April 2016

Fire Containment Drone

Bryan Anthony Toribio
Worcester Polytechnic Institute

Cameron Duross Peterson
Worcester Polytechnic Institute

David Parker Rubenstein
Worcester Polytechnic Institute

Timothy Philip Neilan
Worcester Polytechnic Institute

WZMP:15A3

Fire Containment Drone

April 27, 2016

Team Members

Cameron Peterson *cdpeterson@wpi.edu*
Timothy Neilan *tpneilan@wpi.edu*
David Rubenstein *dprubenstein@wpi.edu*
Bryan Toribio *batoribio@wpi.edu*

Advisors

W. R. Michalson *wrm@wpi.edu*
F. J. Looft *fjlooft@wpi.edu*

Worcester Polytechnic Institute
Worcester, MA

Table of Contents

Table of Equations

Table of Tables

Table of Figures

Section	***Author***
1.0 Introduction	All
2.0 Literature Review	All
2.1 Background on Fire	Cameron Peterson
Basics of Fire	Cameron Peterson
Fire Classifications	Cameron Peterson
2.2 Fire Suppression Techniques and Agents	David Rubenstein, Tim Neilan
Carbon Dioxide as a Fire Suppressant	David Rubenstein
Water as a Fire Suppressant	David Rubenstein
Man Made Chemical Extinguishing Agents	David Rubenstein
Fire Suppressant Foam	David Rubenstein
Halons	David Rubenstein
Fire Suppressant Gels	David Rubenstein
Experimental Methods	Tim Neilan
2.3 Boat Fire Causes	Cameron Peterson
2.4 Need for FCD Support	Tim Neilan
2.5 Drone Technology and Applications	All
Fixed Wing Drones	Tim Neilan, Bryan Toribio
Multi-Rotor Drones	Bryan Toribio
How Fires Affect Drones	Cameron Peterson
Heat of Fire on the Drone	Tim Neilan
Available Commercial Drones	David Rubenstein
2.6 Summary	Bryan Toribio
3.0 System Requirements	All
3.1 System Overview	Bryan Toribio, Tim Neilan
Selection of Drone Platform	Bryan Toribio

Selection of Fire Suppressant	Tim Neilan
System Configuration	Bryan Toribio
3.2 Operational Overview	Cameron Peterson, Tim Neilan
Deployment of Drone and System	Cameron Peterson
Execution of Drone and System	Tim Neilan
Operating Range Based off of Heat Dissipation	Tim Neilan
3.3 Mechanical Design	Cameron Peterson, Bryan Toribio
FireIce Container	Cameron Peterson
Pressurizing System	Cameron Peterson
Temperature Shielding	Cameron Peterson
Spraying System	Bryan Toribio, Cameron Peterson
3.4 Electrical Design	Bryan Toribio, Tim Neilan
Drone Power Distribution	Bryan Toribio, Tim Neilan
Data Transmission	Bryan Toribio
Deployment System	Tim Neilan
Sensor Integration	Tim Neilan, David Rubenstein
3.5 Software Architecture	David Rubenstein
Data transmission	David Rubenstein
System Control	David Rubenstein
System Control Interface	David Rubenstein
Temperature Sensor Reading	David Rubenstein
4.0 Final Design	All
4.1 Mechanical Final Design	Cameron Peterson
Spraying System	Cameron Peterson
4.2 Electrical Final Design	Bryan Toribio, Tim Neilan
PCB Design	Bryan Toribio, Tim Neilan
4.3 Control System Final Design	David Rubenstein

4.4 Software Final Design	David Rubenstein, Tim Neilan
5.0 Results	All
5.1 Deployment System	Cameron Peterson
Diaphragm Pump	Cameron Peterson
Nozzle Adapter	Cameron Peterson
Nozzle Housing	Cameron Peterson
L-Brackets	Cameron Peterson
FireIce Tubing	Cameron Peterson
5.2 PCB Design	Bryan Toribio
5.3 System GUI	David Rubenstein
5.4 Subsystem Control	David Rubenstein
5.5 Sensor Array	Tim Neilan
6.0 Conclusions	Cameron Peterson
7.0 Recommendations for Future Projects	Bryan Toribio, Cameron Peterson, Tim Neilan
7.1 Mechanical	Cameron Peterson, Bryan Toribio
7.2 Electrical	Bryan Toribio, Cameron Peterson, Tim Neilan
7.3 Software	Bryan Toribio, Cameron Peterson, Tim Neilan
Appendix A	Tim Neilan
Appendix B	David Rubenstein
Appendix C	Bryan Toribio
Appendix D	Cameron Peterson
Appendix E	David Rubenstein
Appendix F	David Rubenstein
Appendix G	David Rubenstein

Acknowledgements

As a team we first would like to thank the Fire Protection Engineering Department, specifically, Professor Puchovsky and Professor Dembsey for helping us to understand the problems associated with firefighting and fire dynamics. Both of you were extremely helpful in pointing us in the right direction for our research and who to interview.

We would like to thank the following Fire Deparments: Newport Fire Department, Portsmouth Fire Department, and the Worcester Fire Department. These departments helped us gather real world information on what firefighters would like to see in a drone firefighting system.

We would also like to extend a huge thank you to Gerry Kennedy, Mike Matros, Matt Struzziero, and Ed Kleiman from Geltech Solutions for all of their help providing us with their experience and FireIce for us to test our system. Without all of you the project might not have turned out as well as it did.

The team would also like to extend a thank you to Joe St. Germain, for all the help not just with the project, but also for all the help getting us to this point in our college careers. You provided us with a place to work as well continual assistance when we encountered roadblocks in the project.

Finally, we would like to thank our advisors Professor Michalson and Professor Looft. You both played an integral role in the development of our project. The constant support and guidance from the beginning helped us create this project that will hopefully be continued to become a fully tested and perfected system.

1.0 Introduction

In 2012 there were approximately 268,000 deaths caused by fire related incidents worldwide (WHO, 2012). Within the United States, there were 1,298,000 fires reported in 2014 (WHO, 2012). These fires caused 3,275 civilian deaths, 15,775 civilian injuries, and $11.6 billion in property damage (Hylton, 2015).

According to the National Fire Protection Association (NFPA), fire is a chemical reaction caused by the rapid oxidation of a substance (NFPA, 2015). The base substance can vary from wood, plastic, or even metals. As the process begins, fire releases light and thermal energy creating what we know as fire (NFPA, 2015). In order for fire to occur, there needs to be three components present: a fuel source, heat, and oxygen. These three components are known as the Fire Triangle, shown in Figure 1[1]. This is the basis for understanding how fire operates.

Conventional means of fire suppression use an understanding of the composition of fire shown in the Fire Triangle to extinguish fire. The goal of fire suppression tactics is to try to eliminate one of the three sides of the triangle. The most common means of extinguishing fires begins with the use of water. This is due to waters abundance and ability to smother the fire and dissipate heat. Other forms of fire suppression include the use of foams, gels, and fire blankets. Fire suppressants all work to eliminate the threat of fire and prevent the fire from reigniting.

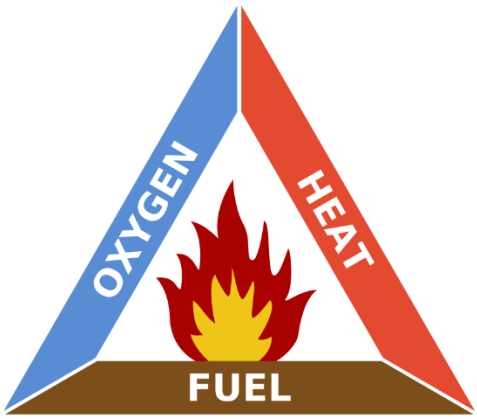

Figure 1: Fire Triangle

The size of a fire can range from small residential fires to widespread forest fires. Any fire can quickly spread due to the introduction of more material to burn or a rush of oxygen. For example, forest fires can spread in dry environments where the vegetation is more susceptible to catching fire.

Fires on remote platforms, for example on boats, are particularly problematic because the fires tend to be isolated, difficult to respond to quickly, and the location of the fire might preclude the occupants from safely evacuating the area. When a fire becomes too dangerous for the boat operators to control, fighting the fire requires outside support from fire departments. Similarly, to fighting a fire on land, boat fires are primarily fought by pumping water onto the fire.

Firefighters are often the ones on the front line when dealing with fires, putting their lives at risk in order to save others. In 2013, 106 firefighters died while on duty in the United States (FEMA, 2015). Unfortunately, the protective gear that firefighters wear can be cumbersome, and can pose additional risk when fighting boat fires, due to the weight of equipment and risk of falling in the water. As a result, a better approach to assist in fire suppression should be developed that reduces or eliminates the need to put people in danger.

A solution to eliminating the human factor in remote platform firefighting and to potentially improve response time would be to utilize drone technology. In 1955, fire departments started to use platforms

[1] Image citation: Wikimedia, “Fire Triangle”, https://en.wikipedia.org/wiki/Fire_triangle

such as planes and helicopters to suppress forest fires (Sargent, 2011). Firefighting planes carry a payload of some fire suppressant, and drop the payload on the fire. Drones, either fixed-wing or multi-rotor, have been used to aid in fire suppression by providing surveillance for the fires. This becomes a valuable resource when dealing with remote fires, for example, fires on boats. The next step would be to have the multi-rotor drone use a fire suppression system while also providing the quick response for surveillance.

1.1 Project Statement

The goal of the Fire Containment Drone (FCD) project is to develop a prototype drone system that is designed to specifically aid firefighters in containing and suppressing boat fires.

To accomplish this goal, we investigated different fire suppression systems to determine what would be the best approach to combat boat fires. Subsequently, the system design adapted existing drone technology to aid in designing a better solution for further development. Additionally, there currently exist aerial systems that deploy fire retardants onto the fire that apply to this project. This project includes further research into finding ways to modify these systems to be the most effective in boat fire scenarios.

1.2 Discussion

A common strategy when fighting fires is to attack the base of the fire, meaning attacking the fire at the source. (Norman, 2012) Attacking a fire at the base is more effective but firefighters often cannot reach the base of the fire because of heat, size of the flame, and risk of death. Using the FCD, however, firefighters could fly the drone into high-risk situations that are typically too dangerous for firefighters.

The primary scenario that this project will focus on is surface fires on boats. The reason for this direction is a boat deck fire will allow the FCD ample maneuverability room as compared to an enclosed environment. The target boat sizes will be 35-40ft boats. The scale of these boat fires is reasonable for the size limitations of drones compared to that of large forest fires.

To slow the spread of the fire, there are requirements that the system needs to meet in order to successfully slow the fire. The first of these requirements is on the drone system. Since a harbor master or firefighter is supposed to deploy the system from the harbor, there needs to be a requirement on how far the drone can travel and how long it can stay in the air. When the FCD is in the air, there are also requirements as to how the drone needs to deliver the fire extinguishing agent. The fire can also present a problem when flying because of the turbulence that the heat creates. To make sure that these requirements are followed, there also needs to be an array of sensors to verify that the system is operating correctly.

Another important requirement is having the ability to effectively suppress the fire on the boat. Considering that boats typically run on diesel fuels and contain electronic systems, the assumption can be made that the fires will be of Class A and B. As a result we will need to develop a suppression system that can effectively handle Class A and B fires. Fire Classes are explained in further detail in Section 2.1.2 Fire Classifications.

With a variety of fire suppressant options available, there will need to be research done on the most effective fire suppressant for boat fire scenarios. Choosing the suppressant will be the foundation of how we develop the system to deploy and apply it to a fire. There will have to be design considerations to fulfill the mechanical, electrical, and software design for the entirety of the drone system. The next sections will delve into the different design considerations that will go into the overall system.

The mechanical design of the drone system will have specific features to fulfill the requirements set above in the Project Statement. In order to deliver the extinguishing agent to the fire effectively, there will need to be a deployment system to launch the agent. To fulfill this requirement, there are a variety of pumps and application systems that can be adapted to our system.

In addition to developing a method of deploying the agent, the mechanical system needs to apply the agent onto the fire. There are a variety of nozzle and spray systems in preexisting fire extinguishing systems that can be adapted to fit our system. Other considerations include incorporating the mobility of the drone to provide unique motion to the application of the agent in order to cover a larger area.

In order to remain mobile, our system needs to have a method of containing the extinguishing agent. How to contain the agent will depend on the type of agent that is best fit for the project. With the variety of fire extinguishing agents including liquids, foams, powders, and gases, the container must be able to effectively hold the retardant. There will be different design considerations if the extinguishing agent needs to be pressurized or not.

Furthermore, the FCD will have several electrical requirements to satisfy the needs of the drone and the chosen fire extinguishing system. The whole system needs a portable power source, such as a large Lithium Polymer (LiPo) battery. The battery will need to sufficiently power all subsystems of the drone. There will be tradeoffs that will need to be considered, such as flight time provided by the battery versus the physical weight of the battery itself. With the limited payload of drones, the size of the battery will determine the operating ranges of the drone system.

Considering the size of the battery that is intended to power the entire drone system, there will need to be an effective means of distributing power to the smaller subsystems. Assuming that the subsystems will require less voltage and amperage, there will need to be high efficiency DC/DC converters to power the subsystems. With the development of each subsystem, there will need to be electrical analysis of each system to determine the appropriate operating ranges.

An important subsystem for the drone system will be the array of sensors. There will need to be a way to verify the ideal operation of the drone when in flight. Since the drone system is intended to combat fires, sensors to detect heat will be useful. One of the specific sensors that is needed is the camera. This is needed to provide a first-person view in the perspective of the mechanical system for the pilot.

The control of the drone will also need to be considered in the design our system. There are a variety of flight controllers tailored to drone systems. The flight controller will need to provide the user with a variety of sensor data and flight log information. The flight controller will also need to allow for the incorporation of additional sensors and features as well as the ability to make modifications so that the flight controller can be tailored for the drone application.

In addition to the control of the drone is the ability to allow user input. There will need to be a way to control the drone as well as a way to provide user input to control the different subsystems of the drone. The drone system will also need to transmit sensor data to the user at the ground station.

To address this there are wireless transmission systems that can be adapted for the project. The purpose of these transmission systems is to provide an effective way to provide communication to and from the

drone system. When choosing these systems, considerations will include the range of transmission as well as the operating frequencies of each wireless transmission systems.

With the variety of sensors and data collection, there will need to be a way to process all of the information. As a result, a micro-controller is necessary to handle all of the data as well as the software that runs on the controller. The controller will need to be able to process sensor data and provide the data to the user in a readable format. There will also need to be further conditioning of sensor data to perform additional tasks like provide warnings to the user.

With the data processed from the proposed logic controller, there will need to be a way to provide user input as well as a way to display the data. A solution to this would be to develop a Graphical User Interface (GUI) that can contain different ways for the user to input commands and visual outputs for sensor data. This solution would be utilized at the ground station for the user.

1.3 Summary

For the remainder of the paper, we explore the science behind fire and effective methods to extinguish fires. The intent of this project is to use this research to adapt a multi-rotor platform to combat surface boat fires. We will delve into the design of the fire extinguishing system developed to address the goal of this project.

Figure 2 below shows the proposed configuration for the project. The diagram is intended to give a high level view of all components of our system we plan to explore within the project.

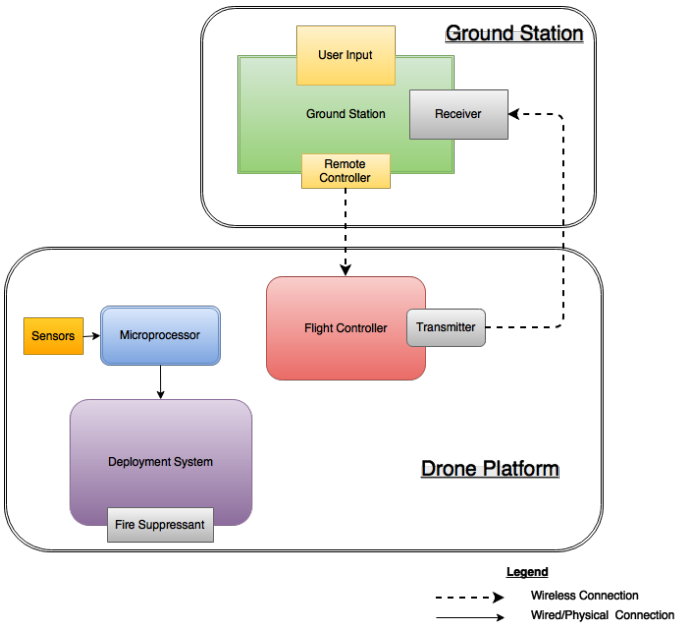

Figure 2: High Level System Block Diagram

2.0 Literature Review

2.1 Background on Fire

These following sections are written to provide a basic understanding of the dynamics and design challenges that arise when designing robotic systems and firefighting systems. Topics covered will include the basic methods of fighting fires, current firefighting technologies, and of course the physics and dynamics of fire.

Basics of Fire

Fire is the release of energy at an extreme rate that produces light and heat (NFPA, 2015). To achieve combustion, three basic components are needed to start and sustain a fire: fuel, heat, and oxygen. The combination of all three of these composes is what is known as the Fire Triangle as shown in Figure 1 above. If any one of these is removed or not present, then the triangle is broken and the fire is extinguished or simply not possible.

Most methods of firefighting focus on denying one of these components to an existing fire, but other methods affect multiple sides of the Fire Triangle. For example, the intent of using a fire blanket is to smother the fire by removing the oxygen, but does nothing about the heat or the fuel for the fire. When firefighters use water, however, the water both smothers a fire and dissipates the heat from the fire.

The fuel component of fire gives the fire energy, which is released and turned into light and heat. Fuel for a fire can range from clothing and chemicals, to building materials or even metals. Each fuel has different traits that affect the way the fire should be extinguished. For example, when extinguishing a campfire, all a firefighter needs is a bucket of water. If firefighters tried the same tactic on a similarly sized grease fire, the water would cause the fire to explode and spread. (Rocky Mountain Fire Department)

Since fire can behave differently based on fuel, fire is broken up into different classes, based on the fuel source. This is discussed in greater depth below in the FIRE CLASSIFICATIONS subsection. The fuel source of the fire plays an important role in how much heat is transferred by the fire to the surroundings.

Heat is the transfer of thermal energy from one body to another. There are three different ways that heat is transferred: *radiation*, *conduction*, and *convection*. *Radiation* relates to the direct transfer of energy between surfaces through a vacuum. Energy is transferred through photons at a wavelength of 1 to 100μm (SFPE Handbook, 1-73). *Conduction* is the transfer of heat energy through a medium, which can be any liquid, gas, or solid (SFPE Handbook, 1-73). *Convection* is a combination of conduction and the natural, or forced, movement of the medium, usually air. Since a fire causes energy to move through the air, the air must also move causing turbulence. In addition to the turbulence generated from fire, there is also the oxygen that fuels the fire that is continuously replaced due to moving air currents.

The oxygen in the air provides an essential part of the chemical reaction that is fire. The chemical equation for each fire depends on the chemical composition of the fuel burning, but as stated before oxygen is usually not considered the limiting reactant. For example, $C_3H_8 + 5O_2 \rightarrow 3CO_2 + 4H_2O$ is the balanced chemical equation for a propane fire. Without the oxygen, the reaction cannot happen. There are other chemicals in air that need to be taken into consideration in the reaction, but since oxygen is the

element that perpetuates the fire, the nitrogen in the air is just carried through the equation (SFPE Handbook, 1-92).

Fire Classifications

Every fire can be categorized into one of five classes of fire, either Class, A, B, C, D, or K (NFPA, *All About Fire*). Each of these is briefly described below.

Class A fuels are ordinary combustibles such as wood, cloth, and some plastics (NFPA, *All About Fire*). Most standard fire extinguishing methods will work on these fires since the fuel does not have high amounts of energy or explosive chemical reactions.

Class B fuels are comprised of flammable liquids such as: petroleum, grease, and alcohols (NFPA, *All About Fire*). Water may not extinguish *Class B* fires because the water is often not effective enough at cutting the fire off from the oxygen. Foams[2] are usually used to extinguish *Class B* fires because of their increased effectiveness at eliminating the oxygen.

Class C fuels are the same as for *Class A* or *B*, except that it has an electrical component energizing the fire (NFPA, *All About Fire*). To combat *Class C* fires usually CO_2 is used to displace the air from the environment to smother the fire. Additionally, by negating the source of electricity from the fire, a *Class C* fire will become a *Class A* or *B* fire, and can be handled accordingly.

Class D fuels are comprised of combustible metals, such as: magnesium, sodium, and lithium (NFPA, *All About Fire*). Traditional firefighting techniques such as using water and foam will not work because of the temperature. To extinguish *Class D* fires, a powder-based chemical must be used such as sodium chloride or other salts, as metal fires can burn so hot that water would break down into hydrogen and oxygen and fuel the fire further (NFPA, *All About Fire*).

The last fire classification is *Class K*, which encompasses cooking media such as: vegetable or animal fats and oils (NFPA, *All About Fire*). *Class K* fires can be extinguished using similar methods to *Class B* fires, but with different chemicals for the foam.

2.2 Fire Suppression Techniques and Agents

There are many commonly used techniques used to suppress fires. These include carbon dioxide, water, foaming agents, surfactants, wet and dry chemical mixtures and additives such as FireIce (Conroy, 17-1). There are also some experimental methods, such as disrupting the flame with low frequency sound waves.

Carbon Dioxide as a Fire Suppressant

Carbon dioxide (CO_2) has been in use for as a fire suppression system for at least a hundred years (Wysocki, 17-3-17-5). It can suppress fires in most flammable materials, except for those such as combustible metals that contain and produce their own oxygen for combustion. As a gas, CO_2 can spread out and blanket a fire, separating it from the air to starve it of oxygen and extinguishing it. The density of CO_2 is approximately one and a half times that of air, allowing it to push the air out of the way (Wysocki,

[2] Foams are discussed more in depth in Section 2.4.3

17-3-17-5). CO_2 only has a small effect on the temperature of the burning material, except when applied through a fire extinguisher directly to the source of the fire (Wysocki, 17-3-17-5).

Water as a Fire Suppressant

According to National Fire Protection Association (NFPA) Fire Protection Handbook, water is the most widely used and most readily available fire extinguishing agent, as it is able to attack all three sides of the fire triangle, the fuel, the oxygen, and heat. Most cities and towns have some sort of fire hydrant system, allowing for easy access to a source of water for use in fighting fires. Unlike some gaseous agents, water can safely be applied to fires in occupied buildings, as water's liquid form reduces the chance of asphyxiation of occupants.

Water reduces heat of the fire, as it has a higher specific heat compared to other materials, allowing it to absorb an incredibly high amount of heat in an attempt to reduce the fire below water's ignition temperature. The specific heat of a substance is defined as the energy required to raise the temperature a certain mass of the substance, usually measured in Joules per gram. Additionally, water also has an extremely high latent heat of evaporation, meaning it can absorb an extremely large amount of heat during evaporation. When working with a liquid fuel, adding water can dilute the fuel, spreading it out, and having less fuel at any part of the fire. Finally, as a liquid, water is denser than air, displacing the air from the fire, and starving it of oxygen.

Man Made Chemical Extinguishing Agents

There are two types of general chemical suppressants, dry and wet. Dry chemical agents are powders often consisting of sodium bicarbonate, potassium bicarbonate, or ammonium phosphate, as well as additional added particles to provide resistance to moisture, and allowing for proper flow from an extinguisher or other application method. The dry agents are typically used for *Class B* and *C* fires, however, ammonium phosphate based agents are sometimes also used for *Class A* fires. Dry agents primarily work by interfering with the reaction causing the combustion. The extinguishing effect of dry chemical suppressants does not last unless sources that could cause re-ignition such as extremely hot surfaces are removed from the area. (Lake, 17-17)

Wet chemical agents usually consist of some sort of salt mixed with water. Common salts used include potassium carbonate, acetate, citrate or some combination of the three. The most common uses for wet chemical agents is for extinguishing fires in commercial kitchens. In the past dry chemicals have been used in kitchens. With many kitchens switching to using vegetable oil instead of animal fat, however, dry agents are no longer effective due to the lower ignition temperature of vegetable oil. This in turn causes the fire to reignite more easily, rendering the dry chemicals ineffective. Wet agents are primarily distributed via fixed distribution systems, however, they are occasionally also distributed via fire extinguishers to support the fixed systems. The primary downside of wet chemical agents is they are not usable in situations where there is energized electrical equipment. (Lake, 17-24)

Fire Suppressant Foam

Fire suppression foams are created using a concentrate mixed with water, and then aerated to create foam (Scheffey, 17-45). Foams are used to suppress liquid fuel vapors as well as cooling the surface of the liquid below its auto-ignition point. The auto-ignition point of a fuel is when the temperature at which a

fuel will spontaneously combust without any triggering mechanism such as a spark (Scheffey, 17-45). When spraying the foam, firefighters blanket the surface of the fire to smother the fuel source. According to the NFPA handbook, foam is the only permanent extinguishing agent used for spills or tanks of burning flammable or combustible liquids. (Scheffey, 17-45)

There are a number of different types of common foams, including aqueous film-forming foams (AFFF), fluoroprotein (FP) foams, film-forming fluoroprotein (FFFP) foams, and protein (P) type foams. Varieties of these foam categories, including low-temperature agents, medium and high expansion agents, and alcohol-type agents are also available for firefighting. Low-temperature agents are used in extremely cold settings, while medium/high expansion agents are used when trying to control a fire by flooding the entire room that contains the fire. Alcohol-type (AR) agents are used for fires that involve fuels that are water soluble or similar. This includes common hydrocarbon fuels that have had an extremely small percentage of ethanol mixed into the solution. The AR type agent keeps the foam from decomposing when they are exposed to the substances that would break down most types of foam. (Scheffey, 17-46 - 17-47)

Another subset of foaming agents is surfactant foams. Surfactant foams are commonly used as wetting agents, however they can also be used similarly to any of the other types of fire-fighting foams when formulated properly. A surfactant works by reducing the surface tension of the water, allowing it to spread out over the fire to choke out its oxygen access more easily. (Scheffey, 17-48)

Halons

Halons are a type of hydrocarbon where some of the hydrogen atoms are replaced by atoms that are part of the halogen family of atoms. These atoms include fluorine, chlorine, bromine, and iodine. This chemical replacement creates a gas that has ideal fire extinguishing properties, especially when used in total flooding systems. Traditionally the extinguishing agents need to break a side of the fire triangle, but Halons add a new way of extinguishing fires. Halons impede the chemical reaction needed to sustain fire, thus putting out the flame. The major problem with using Halon extinguishers is that they have been identified as the most potent of all ozone-depleting substances. (DiNenno and Taylor, 17-93)

Fire Suppressant Gels

FireIce is a potassium-based polymer that mixes with water to create a gel. FireIce is non-corrosive, and it has a freezing point of 27°F when mixed with water. FireIce has a *Class A* fire suppression approval from Underwriters Laboratories, a global safety standards organization. (Underwriters Laboratories, 2015)

As a gel, FireIce sticks to most surfaces that it is applied to, allowing it to protect vertical surfaces because it does not fall to the ground after application. A gaseous or liquid chemical would have trouble protecting a vertical surface, due to the flow properties or dissipation of the liquid or gas. FireIce has been primarily used in wildfire suppression applications. According to a press release from GelTech Solutions, FireIce has been used on over 600 air tanker missions in North America this past wildfire season (GelTech Solutions, 2016). According to Gerry Kennedy, a representative for GelTech Solutions, FireIce works by reinforcing the water molecules, significantly increasing its fire resistant properties. In addition, according to Gerry Kennedy, FireIce is one of the only chemicals that is effective for extinguishing lithium battery fires, as other suppressants react badly with the burning lithium (Kennedy, 2015).

When mixed using salt water instead of pure water, however, the salt disrupts the polymer, causing the FireIce to degrade and become unusable. If FireIce were to be used in boat applications, it would need to be mixed ahead of time, and transported out to the fire as a premixed dissolution, rather than being mixed on scene.

Experimental Methods

In the effort to find new ways of extinguishing fire, several experimental methods have been developed and studied to see their benefits and practicality. One of the major advances of these experimental methods are a result from the Defense Advanced Research Projects Agency (DARPA) and their Instant Flame Suppression (IFS) program. The program was created in order to develop a system that would rapidly extinguish a fire. The applications for this program were directed at fires in enclosed spaces that are difficult to reach, such as large ships' below deck rooms. The IFS program advanced research into two different methods of experimental fire suppression, acoustic waves and electrostatic and magnetic waves (DARPA, 2008).

From DARPA's report and experimentation, it was found that acoustic waves were able to put out small-scale fires in isolated areas (DARPA, 2008). The system that they developed to perform the experimentation consisted of two large speakers on both either sides of the small-scale fire. The introduction to low frequency acoustic waves agitates the surrounding air around the fire, eliminating the fire's source of oxygen. In 2014, engineering students Seth Robertson and Viet Tran at George Mason University managed to make a working fire suppression system using acoustics. Their prototype was significantly smaller than the DARPA version, portable, and able to extinguish small-scale, controlled flames (Brauer, 2015).

Additional research has been made to assess acoustic waves as a fire suppressant in space and microgravity applications, as traditional fire suppressants create additional issues when used in space. Recent studies have indicated that sound waves are capable of extinguishing fires in microgravity more effectively than in regular gravity, although only small scale tests have been performed (Biesner, 2015).

Research from all sources indicates that lower frequencies displace the flame further. The benefits of using acoustic fire suppression would be that it would work on multiple types of fires and would be able to continually suppress fire without running out of a fire suppression agent. Acoustic waves, however, do not remove heat effectively from the area so the chance of re-ignition is higher and there has not been a system developed for larger scale fires.

The other experimental method that was developed by DARPA's IFS program was applying electric fields to the source of the fire. From DARPA's final report, the results indicated that by generating a large electric field around 35kV and directing it at a flame, the flame could be deflected at an angle determined by the application area and magnitude. The results from this report show that an electric field is able to affect a flame, although it is not a practical approach given the needed magnitude and limited effect on the flame.

Many applications of electric fields, and even magnetic fields, were explored such as varying different levels of electric fields to affect flames through and around physical barriers as well as creating areas where fire could not spread. This experimental method, however, was inefficient compared to traditional

fire suppression techniques and has not been expanded to fires larger than small fires used in experimentation (DARPA, 2008).

2.3 Boat Fires

Fires that occur on vehicles can be extremely deadly to people because of the restrictions on egress. Specifically on boats, the restrictions on egress are even more severe. The water surrounding the boat may seem like it can be easily used to extinguish fires on boats; however, the challenge for passengers trying to suppress the fire is that they cannot access the water easily while remaining on the boat.

Fires on boats start from a couple of main sources. These sources can be found in Table 1 below, with their frequency of occurrence (Leonard, 2015).

Cause of Fire	% Total Boat Fires
AC and DC Wiring/Appliance Issues	55%
Engine/Transmission Overheat	24%
Fuel Leak	8%
Miscellaneous	7%
Unknown	5%
Stove	1%

Table 1: Causes of Boat Fires

As seen in the above table the main cause of fires on boats is the electrical wiring. The electrical problems that cause these fires are the wires chafing which can spark starting the fire. Another source of the boat fires is the AC power that is run to the boat from shore. The AC power lines from the shore can be attached incorrectly or not fully attached, which can spark, increasing the risk for fire. (Englet, 2013)

The main challenge from an electrical fire, or *Class C* fire, is the potential to reignite even after the fire has been extinguished. Re-ignition can be a result of improperly shutting off the electrical power source or heat from the previously burning material that has not reduced. (Englet, 2013)

2.4 Need for FCD Support

A multi-rotor drone could assist many ways in firefighting, specifically with boat fires. From first-hand accounts by Michael O'Brien, Deputy Chief of the Portsmouth Fire Department and Kelley Brown, a Fire Protection Engineering student with experience in the Coast Guard, we determined some concerns with boat fires that are encountered by responders. These problems included locating the boat on fire, responding quickly enough, providing life safety, and suppressing the fire while providing egress for the people on board.

Current applications of drone technology in fire protection are predominantly for surveillance (Chu, 2014). A drone could potentially help locate a boat fire with an array of specialized sensors and cameras. Additionally, by flying the drone at higher elevations the user's line-of-sight is increased, and would therefore be able to detect the fire at a greater range. The use of a multi-rotor drone would also provide an additional scouting tool as it could be deployed in a different direction than the responders and expand the search area covered. Deploying multiple drones would increase the efficacy of the search. Once a multi-

rotor drone has located the fire, the drone can lead responders to its location by use of an onboard beacon or GPS location.

The use of a drone would help to further reduce response time, a crucial factor in firefighting, as well as the rescue of people on board the boat. As soon as the distress call from the boat is made, an already prepared drone could be deployed from shore or on a responding boat and sent to the distress location, or if no good location is given, then the drone can be deployed in the general direction. The drone can be deployed in advance of responders, which as previously discussed, would allow for a faster assessment of the area, and a wider area of coverage for surveillance.

Life safety is the primary concern of first responders. In the situation of boat fires, one major concern is that passengers of the boat may need to jump overboard into the surrounding water to flee the fire of the boat. In doing so, the people now trade the danger of the fire for the danger of the water. The two major concerns of being in the water are drowning and hypothermia.

A multi-rotor could aid responders in is the suppressing of the fire itself, and providing a path of egress for those trapped onboard by extinguishing parts of the fire. With the ability to find and get to the boat fire faster, the drone can engage the fire before it grows. The drone then could work to actively contain the fire from spreading and becoming more of a threat. Additionally, the FCD could assess the fire and identify if there is anybody onboard. If there are people onboard, the FCD can allocate its fire suppression payload to make sure a safe path of escape is available.

2.5 Drone Technology and Applications

A drone, often known as an Unmanned Aerial Vehicle (UAV), is described as an aircraft controlled without an onboard pilot. They may be either autonomously flown or remotely controlled.

Drones have found useful applications in the military. The ability to fly an aircraft without a pilot has been able to improve efficiency and safety in such roles as surveillance and even precision strikes. US military drones are around the size of small planes and in some cases are even larger. In recent years, drone technology has expanded to private and public applications, focusing on smaller drones and other scenarios. Private and public research laboratories have contributed to this development. Furthermore, the recent miniaturization and cost reduction of electronics has also contributed to the size reduction of drone development (Floreano, 2015).

There are two main categories of aerial drones: multi-rotor and fixed wing drones. Multi-rotor drones consist of several two or three blade propellers powered around the center of the drone. Multi-rotor drones have the capability to hover and have potential for precise operations and continued fixed area monitoring. Fixed wing drones are comprised of a rigid wing structure, which generates lift from the drone's forward propulsion system, which is generally a propeller. Fixed wing drones are useful for covering large areas for surveying as well as transporting heavier payloads. Current small drones are being used for surveillance in government agency roles while hobbyists and photographers use them to capture pictures and videos from the sky. In firefighting applications, drones are used primarily for surveillance and monitoring of fires.

Fixed Wing Drones

Fixed wing drones are one of the main categories of drones. They operate similarly to a plane in which the aerodynamics of the wing generates upward thrust because of the forward propulsion. Control of a fixed wing drone relies on mechanisms in the wing. These mechanisms are ailerons, an elevator, and a rudder. The ailerons control the roll of the aircraft, the elevator controls the pitch of the aircraft, and the rudder controls the yaw of the aircraft. *Roll* is denoted as the rotation about the x-axis of the center of gravity (COG) of the multi-rotor drone. *Pitch* is the rotation about the y-axis of the COG of the multi-rotor drone. *Yaw* is the rotation about the z-axis of the COG of the multi-rotor drone. All of this is illustrated in Figure 3.

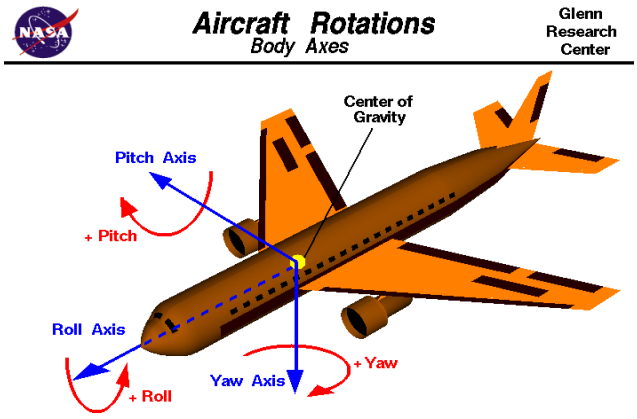

Figure 3: Aircraft Axes of Rotation

Fixed wing drones have been used for a number of applications, civilian, commercial, and military. Since fixed wing drones are of a simple design and do not require an onboard pilot, they are a relatively cheap option for several aerial applications. They have been used in surveying areas and mapping regions. In agriculture, they have been used to monitor and assess large crop fields.

Multi-Rotor Drones

The other main category of drones is multi-rotor drones. Multi-rotor drones are classified as rotorcrafts that operate on more than two rotors. They operate similarly to a helicopter in that they are able to fly due to the displacement of air created by the propellers placed on the drones. The multi-rotor flies upward against the force of gravity with the lift generated from the rotors. With the multi-rotors reliance on multiple propellers, there is a variety of design options based on its designated task. As a result, the characteristic of a multi-rotor design can be tailored specifically to a certain task.

Movement of a multi-rotor is fundamentally different compared to a fixed-wing drone. As stated earlier, the multi-rotor drone relies on propellers to provide lift in order to become airborne. Once airborne, movement of the drone is done through the manipulation of the propellers. We classify the orientation of the multi-rotor using the terms *roll*, *pitch*, and *yaw*.

To move the drone, an onboard computer drives the propellers independently to dictate the movement of the drone. In the case of a quadrotor drone, a multi-rotor drone that has four propellers has complimentary rotating propellers. More specifically, two propellers opposite of each other rotate counterclockwise while the others rotate clockwise. This provides symmetry in the design of the multi-rotor, which is essential for the stability of the multi-rotor. Power is distributed to the different rotors to rotate or translate the drone, allowing it the ability to move with little restriction.

With the maneuverability of the multi-rotor drone also come the characteristics of its design. There is a multitude of different options when implementing a multi-rotor drone. The design process typically begins with the amount of rotors placed on the drone. The amount of rotors is typically in intervals of two

with the exception of a tricopter, which has three rotors. The rationale for having an even number of rotors is to have symmetry on the body of the drone. The amount of rotors on a given drone dictates its capacity to carry and perform tasks. More rotors usually mean that the drone has a higher payload, or ability to lift more weight. There is also the added benefit of more stability with multi-rotor drones that have more rotors.

How Fires Affect Drones

The main ways that fire impacts the drone in this project is how air currents change as the fire changes and the heat that could cause damage to the drone. The air currents created affect the controls and safety of the multi-copter because most drone controls assume a stable environment. Based on the buoyancy of fluids, the hot air generated by the fire creates airflow up towards the drone, causing turbulence (SFPE Handbook, 1-13). Drones are traditionally lightweight and fragile, so turbulence can have a large impact on how well they can navigate.

The safety of the drone is affected by the pilot's ability to navigate in the turbulent air currents caused by the fire, but the heat from the fire can also cause damage to the mechanical and electrical components if they are exposed to too much heat. Most drones have a frame and body made of plastic since it provides a strong enough base and is lightweight, however, plastic also has a much lower melting point than metal which can cause issues if the drone stays around the fire for too long. The electronics and controllers for the drone also have specific temperatures that they must be kept at to function correctly.

Heat of Fire on the Drone

An important characteristic to take into consideration when fighting a fire with a drone is the amount of heat that it is dissipating. The heat dissipated by the fire is influenced by many factors such as type of heat transfer, distance, and time. As previously discussed, the types of heat transfer are convection, conduction, and radiation.

The total heat transfer onto the drone will be the sum of heat experienced by convection, conduction, and radiation. The drone will be flying around the fire so there will be no heat transfer via conduction as there will be no solid contacts made with objects producing heat. In an open-air environment, thermal radiation far outweighs that of convection especially when staying out of the region directly above the fire. Therefore, the primary heat transfer influencing the heat experienced by a multi-rotor drone is radiation.

When considering the amount of heat transferred at a distance, it is important to note the type of material that is the fuel source for the fire as well as the material that the heat is being transferred onto. The material burning will influence thermal radiation because of two primary characteristics, its heat release value, and its radiant fraction. The heat release of an object is how much total heat the object gives off when burning. The radiant fraction is the efficiency of the dissipation of thermal radiation (SPFE Handbook 2002). For the material that is being radiated on, emissivity is a key characteristic to consider. Emissivity is a measure of the efficiency of the surface as a radiator (Drysdale, 1999). Essentially, the emissivity of the material dictates how much of the total radiation at that point is transferred to the object.

Available Commercial Drones

Many companies sell small drones that can carry a camera, or have an integrated camera. These drones, however, do not have significant carrying capacities. There are some companies who produce drones that

have a significantly larger carrying capacity, such as DJI Innovations, whose drones are intended for use in high-end photography and filmography. DJI's signature drone series is their Phantom drones. In January 2014, the Branford, CT Fire Department used a phantom series drone flown by one of their volunteer firefighters to determine if a quarry fire was burning too close to the quarries explosives cache to safely send in firefighters. With the camera on the drone, the fire department was able to determine the fire was far enough away from the explosives, and sent in firefighters to put it out. The Phantom 2 drone, for example weighs 1kg, and only has a takeoff weight of 1.3kg, giving it a maximum payload of only 300 grams.

DJI has a series of drones known as the 'Spreading Wings' series that are their higher payload drones. The S800 has a takeoff weight of 6-8kg, and a total weight of 3.7kg. The S900 has a takeoff weight of 4.7-8.2kg and a total weight of 3.3kg and costs about $1400. The S1000 can carry 6-11kg, and weighs 4.2kg, costing about $1900. Finally, the S1000+ can also carry 6-11kg, and weighs 4.4kg, while costing about $2500. Going even bigger, SABRE, a company specializing in robotic surveying systems for use in inspection and surveillance, produces the HL48 Skyhorse, with a base weight of 6kg, and a takeoff weight of 10-20kg. The downside to the HL48 is its cost, running $15,000 for just the basic drone, without upgrades. High payload drones, though useful, are also extremely expensive.

2.6 Summary

This chapter has presented the foundation of knowledge needed to understand the different components that comprise this project. In order to develop a system to combat boat fires, it is important to understand what fire is and how to eliminate it. Within the chapter, we covered what are the different classes of fire as well as the variety of suppression techniques. We finish with discussing drone technology and its applications. Finally, we combined the fire dynamics with the drone dynamics to review how fire affects drones in flight. The intent of this research was to understand the capabilities and uses of drone technology so that we are better able to utilize a drone for our application.

3.0 System Requirements and Initial Design

3.1 System Overview

The purpose of the project is to develop a fire suppression system to be used on a multi-rotor drone. The drone needs to be capable of having a quick response time as well as being effective when attacking boat fires. To accomplish these goals we discuss what the intended project specifications are as well as the design we plan to use to accomplish these goals.

The system will be comprised of three major components:

- A multi-rotor drone
- The deployment system for the extinguishing agent
- A ground control

To begin, we will first look at the S1000 drone which we had available to use from the RBE program. The S1000 has three major components:

- The drone frame (electronics and motor systems)
- The Pixhawk drone controller
- The drone battery
- The drone sensors
    - Temperature
    - Camera

The Pixhawk controller is a prepackaged system that already controls the flight of the drone, with only minor configuration having been required to fly the S1000.

The deployment system will be modular to allow for ease of detaching the system from the drone for maintenance or use in other applications.

When referring to maintenance, the container will need to be refilled with FireIce. The system will need to allow an easy way to add more FireIce and pressurize the FireIce a desired pressure.

The ground station will consist of the systems necessary to communicate with the drone, such as telemetry communication, video feedback from the camera, information readouts from sensors, and a Graphical User Interface (GUI).

Since many of the specifications of the system are determined by what drone we use, we will now discuss the reasoning for selecting the DJI S1000 as our drone.

Selection of Drone Platform

The drone for the system needed to satisfy the following criteria: sufficient payload, travel speed, flight time, and cost. For our project we defined a sufficient payload to be roughly 10 lbs. This was to take into account the weight of the mechanical system for the project as well as the fire present. A sufficient travel speed was defined at a speed of roughly 30 mph. This was to have a faster response time when deploying

the system. A decent flight time was defined to be over 5 minutes. This flight time would allow sufficient time to deploy the system and apply the fire suppressant.

Using the described criteria above, we compared different drone platforms. Through our research, we determined that a large-scale multi-rotor drone would be the most reasonable platform for our project. Large multi-rotor drones would be able to handle a higher payload, without sacrificing the maneuverability of flight.

From the results of our research, we decided upon using the DJI S1000 Multirotor drone. We found this drone platform to be the most cost effective platform that satisfied all of our criteria. The S1000 is has a payload of roughly 11lb with a flight speed of roughly 40 mph. With a battery of 16,000 mAh, the drone is able to have a flight time of roughly 15 minutes. There were also further benefits of incorporating additional functionality and modifications to incorporate our fire suppression system.

Selection of Fire Suppressant

The decision matrix was used to select the fire suppressing system. The first step consisted of listing fire suppression techniques and agents and ranking their effectiveness from 1 (extremely poor) to 10 (excellent) in four categories: *Feasibility of Application*, *Longevity of Suppression*, *Weight*, and *Fire Containment*. *Feasibility of Application* was defined as how possible the team would be able to easily adapt the fire suppression system onto a flying drone platform given the duration of the project and the resources available. *Longevity of Suppression* was defined as how long and continuously the fire suppression system could effectively suppress the fire. *Weight* of the fire suppression system dictates the functionality and feasibility of the drone's design and performance. The *Fire Containment* category was defined as how effective the fire suppressant system was at keeping the fire from spreading.

The top three techniques and agents were then moved to the second step involving a more in depth decision matrix, where each fire suppression technique and agent were compared directly to each other. The full decision matrix can be found in Appendix A.

From the results of the decision matrix, the best fire suppressant agent to use for our project was FireIce. The most important qualifier for choosing FireIce was its ability to handle Class A, B, and C fires, the most common type of fires on boats. Though FireIce is not yet approved for Class C fires, the solution is electrically nonconductive. As a result deploying on a Class C fire will still prove to be effective. Since fires on boats may initially begin as Class C fires due to electrical issues, they will transition to Class A or B fires as the electrical component is removed. Additionally, most boats are required to be equipped with shunt circuits and other safety precautions that shut off the electrical source of the boat in the event of a malfunction.

Other useful characteristics of FireIce include its ability to stick to surfaces, providing a protective layer. The viscosity of FireIce is helpful for preventing reignition of the fire as well as providing a path of egress when applied. There is also the benefit of having access to samples through the local distributor, which made it a low cost and feasible application to develop.

System Configuration

The system will consist of the deployment system attached to the S1000 drone. There will be a central switch that will provide the power to the motors and control systems. The control systems will contain the following:

- Sensors
- Deployment system
- Video feed system
- Flight Controller (Pixhawk)
- Logic Controller (Arduino)

After turning on the system, there will be a series of checks to determine that the individual systems are working properly. Most of these tests for the individual systems will happen automatically. In the instances of the video feed system the operator will check to determine that the transmitter is properly displaying the telemetry data from the Pixhawk on the LCD display at the ground station. The operator will also check that the remote controller is properly sending the control signals to the drone such that we are able to control the flight.

When the different systems are checked then the drone will be able to be cleared for use. In the next sections we will discuss the drone operation during the different stages of the drone's use. These stages will include the deployment of the system, and the execution of FireIce once the drone has arrived at the desired location.

3.2 Operational Overview

Deployment of Drone and System

Considering the intent of the drone system, there will need to be specifications for proper utilization of the drone. These specifications will detail the proper operation of the drone. Specifically, there will need to be a detailed list of the different flight modes that come with the flight controller. The flight modes will describe how the drone is controlled by the user. The flight modes will also determine the different sensors and subsystems that are used when the drone is being operated. A diagram of the drone operation can be seen in Figure 4.

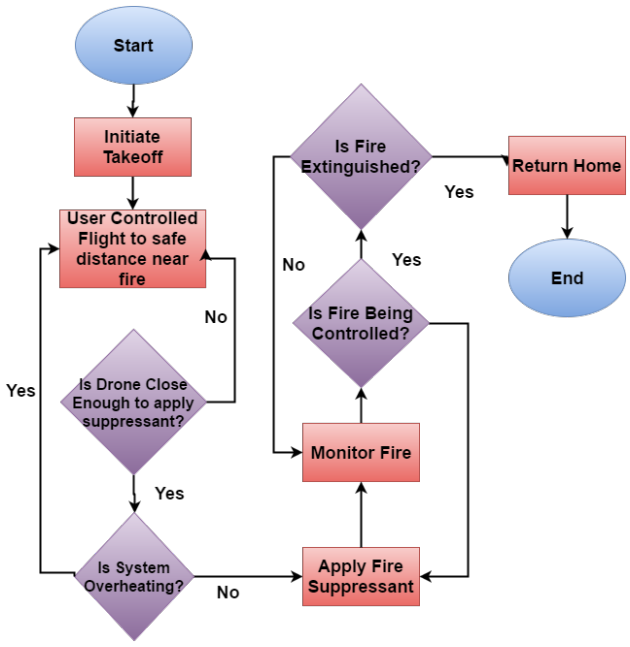

Figure 4: Drone Operation Flowchart

Once the operator has determined that the drone is operational and ready to fly the operator will begin to use the remote control to take control of the drone by putting the drone into *Navigation*

Mode. Once lift off is achieved then the operator can either choose a GPS point for the drone to navigate to and put the drone into *Autonomous Navigation Mode*, or manually navigate the drone. For manual operation the operator can control the Roll, Pitch, and Yaw of the drone during flight with the joysticks on the remote control. The operator can use the GPS tracking of the drone for navigation as well as the camera feed during the period of manual operation. The autonomous mode allows the operator to click a point on the GPS map on the navigation computer; the drone will then travel to the chosen location.

For autonomous navigation, the drone will use an onboard GPS module to provide the current location of the drone. To remain out of danger from ground interference the drone will first fly to 20 meters vertically, above sea level to prevent collisions with other boats during navigation or flying into the sea. Then using an existing point-to-point navigation program the drone will travel to where the drone operator tells the drone to travel, the distance chosen should be about 20m to prevent issues with the drone overshooting the target and flying too close to the fire.

Execution of Drone and System

When the drone has reached the destination through either autonomous or manual control the drone can then be put into *Extinguish Mode*. While in *Extinguish Mode* the drone's manual operation controls sensitivity will be reduced to better maintain accuracy while spraying FireIce. The primary operation of the drone will default to maintain position, using built-in programing of the flight controller.

Two people will handle the control of the drone during Extinguish Mode: the Pilot and the Co-Pilot. The Pilot will be in control of the drone platform. The Pilot is to keep their eyes always on the drone for additional safety. The Co-Pilot will act as the second set of eyes for the Pilot. They will be in control of the remainder of the ground station, having access to the sensor readouts and deployment controls.

During the flight of the drone, the temperature sensor reading will inform the operator of any temperatures dangerous to the drone. The sensor will read a warning when sensing 30°C and will issue a critical warning at 35°C. The temperature sensor will only relay information to the operator, not affect the drone control of the operator. Using the GUI, the user will be able to control the spray of the system. The nozzle will be controllable to provide additional suppressant coverage area. The nozzle will have a clockwise rotation (up towards the drone) and a counterclockwise rotation (down away from the drone). The operator can then choose where the nozzle is pointed and when the FireIce should be sprayed using the GUI further described in section *3.4.3 System Control Interface*.

Operating Range Based Off Heat Dissipation

When controlling the drone there will need to be special attention given to how to safely operate the drone when near a fire. To determine a safe range of operation, there are many variables about the fire that will dictate where the drone can safely operate around the fire. In the next section we will discuss the calculations done to determine the appropriate operating ranges based on the types of fires we will be dealing with.

Our drone has many systems that have to operate within a specific range of temperatures. Our electronics have the lowest maximum temperature failure rating at 40°C. The safe operation of the drone around the fire will depend on the heat dissipated from the fire to our drone.

The heat dissipated by the fire is influenced by many factors such as type of heat transfer, distance, and time. The types of heat transfer are *convection*, *conduction*, and *thermal radiation*. *Convection* is heat transferred through a medium such as the air. *Conduction* is heat transferred through solid contact. *Thermal radiation* is heat transferred from electromagnetic waves (Drysdale, 1999).

For our application and types of heat transfer, only radiation will be a significant factor. Conduction will not be a factor, as the drone will not be making any solid contact with hot surfaces. Convection through the air will be negligible as the drone will operate in an open-air environment and will stay out of the zone directly above the fire where heat convection is most prominent. For our application, we will be considering the steady-state of heat transferred to our drone to determine what position we can hold until the drone is low on power and needs to returns home. Time will not be a factor, as the fire will be assumed to be fully developed during our operation. Therefore, the major contributing factors towards maintaining a safe operating temperature is the amount of heat radiation generated and our distance from the fire. Establishing a relationship between radiation and distance is crucial to the design of our drone system.

In fire protection applications, determining the critical heat flux of a system is vital to accurately modeling the scenarios that would lead to system failures. Heat flux is the amount of thermal energy transferred through the area of a surface over time (SFPE Handbook 2002). Critical heat flux is the amount of heat flux required to cause a system to cease functioning properly. The relationship between heat flux and temperature is not a linear function. Certain materials and applications have found the equivalent critical heat flux and temperature for their systems. In power plants, the critical heat flux for their solid state electronics rated to fail at 65°C was $3kW/m^2$ (NUREG). Thermoplastic cables have a high critical heat flux at $6kW/m^2$ and an equivalent temperature of 205°C. Our system is rated to fail at 40°C, so a critical heat flux significantly less than $3kW/m^2$ will be used for our calculations.

The material that is burning is crucial in determining the relationship of heat radiation over distance as different materials have different thermal and burning properties. The two significant properties of the burning material are heat release and radiant fraction. The heat release is the total amount of heat dissipated by the burning of the material. The radiant fraction is the efficiency of the dissipation of thermal radiation (SPFE Handbook, 2002). These values for certain materials have been gathered and tested experimentally and compiled in fire protection engineering references such as the Society of Fire Protection and Engineering Handbook.

An important aspect of determining the thermal radiation on an object is emissivity. Emissivity is the fraction of how much radiation is absorbed by the object. A low emissivity indicates the object reflects a large amount of radiation. A high emissivity indicates the object absorbs most of the radiation. For our purposes for the drone, we will make a black body assumption, which indicates an emissivity of 1. This means we will assume that our drone absorbs all the thermal radiation directed at the drone. That assumption will provide our calculations some safety factor in determining a safe distance away from the fire.

The model for determining the thermal radiation at a distance from the fire will be a point source radiation model. A point source model assumes the total thermal radiation dissipates from a center point of the fire.

The relationship between the mentioned factors can be shown mathematically for the equation of heat release:

$$Q = \frac{q" * 4\pi r^2}{\lambda_r}$$

Equation 1: Heat Release

Where q" is the critical heat flux, r is the radius from the center point of the fire, and is the radiant fraction. Solving for r, we obtain:

$$r = \sqrt{\frac{Q\lambda_r}{4\pi * q"}}$$

Equation 2: Distance from Fire

Using wood doused with gasoline as a comparable model for our material burning and a critical heat flux of 2kW/m^2, a safe operating distance of 2.82m or 9.25ft was obtained. Using fiberglass, the safe operating distance was 1.43m or 4.70ft. Using acrylic, the safe operating distance was 4.38m or 14.20ft. These materials were chosen as comparable to boating materials that our application would aim to put out as well as materials that we could easily test for. The material properties were listed in the SFPE Handbook. From these calculations and modeling, we specified a safe operating distance of 10 to 15ft.

3.3 Mechanical Design

During the design process for our system, we examined many different ways of solving the problems from our system requirements. The sections below explain each portion of our design and other designs that we considered, as well as why we chose the design we did. There are four main mechanical aspects to the system, the container for the FireIce, the pressurizing system, the temperature shielding, and the spraying system. One of the requirements that we determined specifically for the mechanical system is that the FireIce must be at 100 psi to reach a reasonable distance when fighting the fire. Testing of parts of the system can be found Appendix D at the end of this report.

Figure 5 shows a block diagram of the different components of the mechanical design. We will describe the requirements for each portion, detailing the research and different approaches for section.

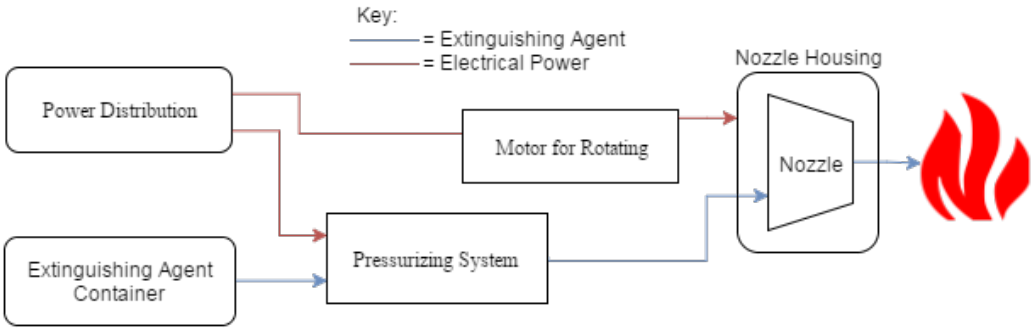

Figure 5: Mechanical Design Block Diagram

FireIce Container

When first discussing our system specifications, we thought that a full gallon of extinguishing agent would be an adequate starting place; however, a gallon would weigh about 8.34 lbs., which would take up most of the 11 lbs. payload of the drone. With that in mind we decided that half a gallon of FireIce would adequately fit within the limitations of the drone platform for the sake of the project. To solve the problem of containing FireIce we explored three different ways to fulfill this requirement.

COTS Container

The Consumer off the Shelf (COTS) container that we look at was the 2 Quart Natural Tank with Mounting Tabs as seen in Figure 7. The COTS container would give us a half-gallon of volume while remaining lightweight as a container since the container only weighs 0.631 lbs. empty. With a full container of FireIce the container would weigh about 4.17 lbs. The COTS container comes with a large opening and with a cap to refill the system with FireIce; however, we would need to add our own connection to attach the COTS container to the rest of the system. This connector would be a barbed hose connector to a ⅜" NPT (National Pipe Thread) connector with a ⅜" NPT female locknut on the container side to secure the adapter. This attachment method is shown in Figure 6.

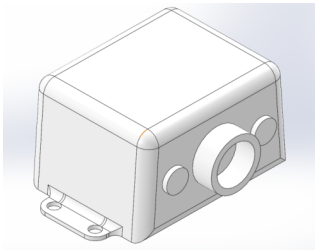

Figure 7: COTS Container

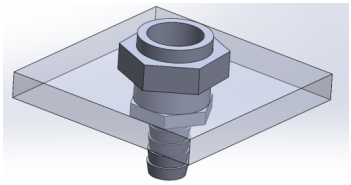

Figure 6: Tube Connector

PVC Container

Another container design that we had considered was a PVC (Polyvinyl chloride) pipe based design, as seen in Figure . The design would use a piece of PVC pipe with 3.5" ID (Inner Diameter) (4" OD (Outer Diameter)) and a length of 6" with custom blocks as feet for support of the pipe made from blocks of ABS (Acrylonitrile butadiene styrene) plastic. These dimensions would give this container a volume of about 0.49 quarts. On either end of the pipe there would be a PVC pipe cap with a ⅜" transport tube adapter.

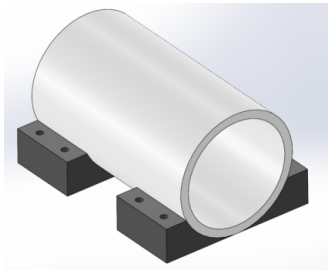

Figure 8: PVC Container

Trapezoid Container

The Trapezoid Container is a design that we developed ourselves using 3/16" Acrylic sheets, a model of the design can be found in Figure . A design challenge that we predicted with most of the containers mentioned previously is that since most of them have flat bases, when drawing FireIce from the container not all of the liquid will drain without extra suction. When designing this container we kept that in mind so we slanted the sides of the container in the design. To further increase the efficiency of the container we added in a sloped plate to direct the FireIce towards the front and bottom of the container where the transport tube adapter is located. This design also incorporates the volume of extinguishing agent that we wanted to carry on the drone; the inner volume of the container is approximately equal to 2 quarts. To have about 2 quarts of extinguishing agent we created

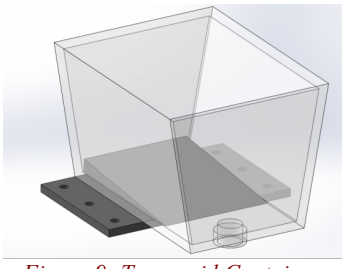

Figure 9: Trapezoid Container

the trapezoid to have bottom base of 100 mm, a top base of 140 mm, a height of 115 mm, and a length of 170 mm. These dimensions do not account for the thickness reducing the volume when determining the volume, so when determining volume the thickness should be subtracted from the dimensions. The internal volume of the container is 1.95 quarts. Given these dimensions the weight of this container empty is 1.35 lbs. and when filled with FireIce the weight is 5.42 lbs.

When determining how to design the container we were concerned that there might be problems with the structural integrity of the container when the drone is flying. To resolve that issue we researched molding Acrylic. The base and the walls can be heat molded to the trapezoidal shape and the top of the container will be attached to the walls with Epoxy. The top of the container will have a hinge that can be locked, to refill the container with extinguishing agent. To further secure this container to the drone we also added a sheet of ABS plastic that will be epoxied to the bottom of the container to attach the container to the drone. Since this container is tall we decided to also design a part that attaches the container to the top of the drone. This design is effective since the container uses the space available on the drone effectively, directs the FireIce into the tube adapter, fulfills our volume requirement, and is relatively lightweight.

Pressurizing System

The second main system of the drone is the system that generates the pressure to spray from the nozzle. One design for this system would be to pre-pressurize a container and have an ON/OFF nozzle to control the spray. Alternatively, we could have a design that could control the pressure by itself and we would not need to pre-pressurize the system.

Motorized Pressurized System

This design used a motor and a worm gear system to drive a plunger into a container to maintain the pressure in the container. A general system mock up can be found in Figure 8. Based on the equipment at the beginning of the section, the system needs to be able to output 100 psi of pressure. Since we did not want to have the motor continually be stalled, we decided that a worm drive gear system would work the best. Even with this worm gear, drive the torque requirement of the motor would be so high that the motor we would need would weigh more than acceptable for the system since weight was such a premium in our system.

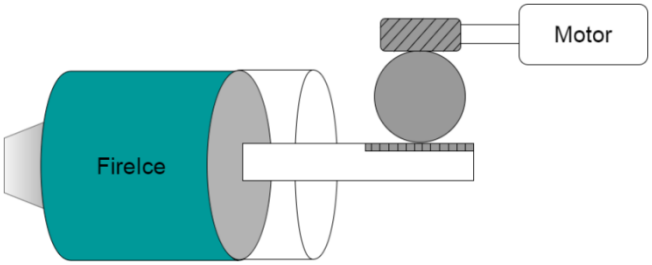

Figure 8: Motorized System

Spring Pressurized System

The spring design was similar to the motorized pressurized system design except this system used springs to apply the force required. The general system mock up for this design can be found in Figure 9 One design flaw that we saw after designing this was that as the system sprayed FireIce the system would reduce the psi of the container since the springs would extend more and more. One way to

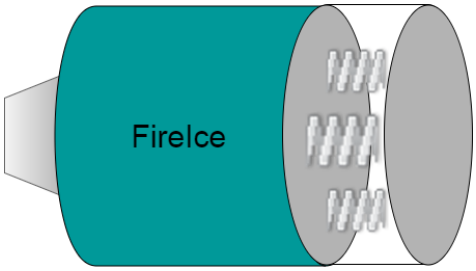

Figure 9: Spring System

overcome this problem was to overcompensate for the pressure at full capacity of the system so that the system would never be less than 100 psi. Even with the overcompensation fix, this design would not work best for our application because the system would need to be pre-charged. Furthermore, the system would be flying around with high-powered springs so if anything went wrong in the system they could damage the rest of the drone. The weight of the heavy springs and the container to hold the springs and FireIce would be higher than what would be acceptable for our system.

Bladder Pressurized System

For this design, we wanted to develop a system that could be easily pressurized with a lightweight mechanism that would not need to be carried by the drone. To accomplish this we developed a bladder that would be able to maintain pressure in the system after being pre-loaded before launch. The mock up for this system can be seen in Figure 10. This system is the lightest design that we developed; however, this system has the same issue that the spring system has, this system loses pressure as the volume of the system is sprayed. The bladder would be placed in the container and pressurized with an external pump. This would give the system the pressure this system needs and reduce the slosh in the container.

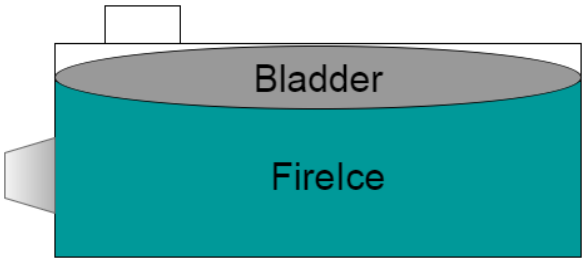

Figure 10: Bladder System

Diaphragm Pump

While researching methods of maintaining pressure in a closed system, we came across diaphragm pumps. While researching diaphragm pumps we saw some potential issues that could arise from by using a diaphragm pump: vibration, inconsistent flow, weight, and pressure. These were the four main concerns we had to make sure that any pump we looked at met. Since Figure 11 is just a mock up the tubing and locations of the FireIce container and the nozzle are not placed where they are on the complete system. We eventually found a pump sold by Estone. This pump had the specifications for what we needed: 1.4lbs, 12V, 60W, 5L/Min, and 115 psi max pressure. All of the physical specifications were within our requirements. This pump allowed us to remove the need for a pressurized container so we could use a much lighter weight container, saving weight and space on the drone. By using this pump, we also could eliminate the need for an electrically actuated valve, further simplifying our system. With this pump, we also can regulate the pressure outputted by controlling the current running to the pump.

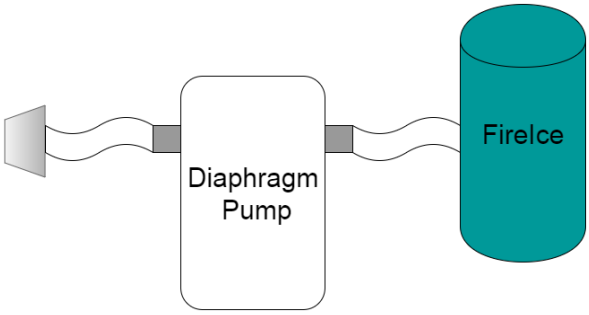

Figure 11: Diaphragm Pump System

Temperature Shielding

As discussed in the Basics of Fire section radiation and convection heat transfer play a large role in the temperature of the drone while the drone is flying around the fire. Since the drone will be flying relatively close to the fire we need to have some sort of protection for the frame of the drone as well as the

electronics on the drone. Once we determined the safe distance based on radiation, we also assumed that the heat transfer from convection would be negligible because of the distance we would be at for radiation. The wind generated from the propellers also would further reduce the heat transfer from convection. Therefore, we focused our design on protecting the drone from radiation.

Aluminum Foil

A low cost and simple solution to add protection to the drone is to tape on aluminum foil to the bottom facing sides of the frame and arms of the drone. Mike Matros from FireIce first proposed this solution during one of our meetings (Matros, 2015). The aluminum foil solution is low cost, has a low emissivity of 0.04, and is lightweight. The issue arises when the Aluminum foil is attached to the drone. Because the foil is not adhesive and is thin, about 0.2mm, the Aluminum foil is extremely difficult to attach to our system. There is the possibility of creating mounting parts or adhesives. The issue with these solutions is that adding an attachment part adds weight and the adhesive solution might not secure all parts of the foil allowing some parts to get caught in the ambient wind or the wind generated by the propellers.

Kapton Tape

Once we started looking at high emissivity materials we found Kapton Tape. This is a type of material designed to reflect the radiation from the fire. The emissivity of the Kapton tape is 0.03, which is not too much of an increase from aluminum foil, but the main advantage is that the Kapton tape is comes with an adhesive backing and is easily attached to any part of the system. Since the Kapton tape is easy to attach and little chance of the tape coming off during a flight, Kapton Tape seems like a good choice to fulfill this requirement

Spraying System

Nozzles

The nozzle for the deployment system is crucial for the application of the FireIce solution. The nozzle will dictate the area of coverage and impact that the extinguishing agent will have. For our purposes, we wanted to choose a nozzle that would best suit the application of the project.

When choosing a nozzle we focused on three major parameters: spray angle, pressure of the system, and viscosity of the solution. Figure 12 shows a graphical representation of the spray angle.

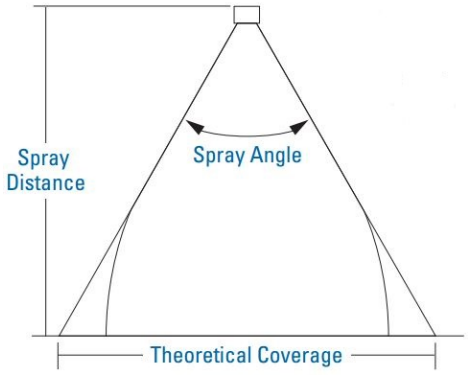

Figure 12: Spray Coverage

The spray angle is the angle at which the solution is sprayed. Larger angles mean that a larger area is covered when spraying while smaller angles are more condensed. Using Spraying Systems, a leading producer of nozzles, as a reference, we were able to gather the data necessary to choose the nozzle that would best fit our application.

Spraying Systems has an expansive catalog detailing their different nozzle and spray systems. Within the catalog there is a technical reference section that details the operating points of different nozzle types. For

this project, we looked into two different nozzle types: tapered flat spray nozzles, and full cone nozzles. These two nozzle types have different spray patterns that would work for the application of the FireIce solution.

According to the technical reference section of their catalog, many of their nozzles are rated based on the spraying of water at 40 psi. The spray angle of a given nozzle is determined by the viscosity of the liquid being deployed, and the pressure in the tube leading up to the nozzle. Through analyzing the data presented by Spraying Systems, an increase in pressure would cause the spray angle of a given nozzle to increase. Inversely, increasing the viscosity of the liquid would decrease the spray angle of the nozzle.

With our application, we will be spraying a solution slightly more viscous than water at a pressure of 100 psi. With FireIce, we have control over the viscosity of the solution by changing the ratio of FireIce powder to water. For the project we plan to keep the solution slightly more viscous than water such that the solution does not compromise the pressurized system. With the solution being pressurized to 100 psi, the increase in spray angle was accounted for when determining the best-fit nozzle for the application.

The next parameter we looked at was integrity of spray coverage. The concentration of coverage degrades the further a nozzle is used away from a target. As a result larger spray angles would have a poor effective area of coverage at large distances. In the case of the data presented by Spraying Systems, much of their data is only rated for distances of roughly 4 ft. With our project, we plan to deploy FireIce at distances between 10 and 15 ft depending on the size and severity of fire. With that, we would need a spray angle that is able to traverse this distance while keeping the integrity of the spray coverage high enough to impede the fire.

Using these characteristics of nozzles, and the parameters for deploying the FireIce, we believe that using a flat spray nozzle at an angle of 15° or less would work the best. With further testing we will be able to determine the best-fit spray angle size for our application. With the knowledge gained from our research we were able to determine that having a spray angle fixed within this range would provide the best coverage and spray integrity for our applications.

Nozzle Motion Control

As discussed before the nozzle of the system will have a spray angle; however, to increase the area of coverage of the nozzle the system should also include a method of rotating the nozzle. In order to realize the motion needed for the additional coverage the nozzle will need to be contained in a housing so that the nozzle will be able to be rotated.

This housing will be attached to the system plate, therefore the plate will cut off 180° of the effective spray area, then the rest of the system will likely be behind the housing cutting off another 90°. Leaving a needed rotation of 0° to 90°. Since this rotation should not be continuous, because of the hose attached to the nozzle, the most effective way to create the motion needed would be through a servomotor. The servomotor would also eliminate the need for a sensor to determine position. The exact position may not be known, but the general location will be and that will be enough for this application.

In order to rotate the housing of the nozzle the servomotor will need to be able to output enough torque to rotate the housing and bend the tubing attached to the nozzle. When designing the spraying apparatus for the system we considered three different servomotors: the Vex 3-Wire Servo, a Micro Motor from Hitech,

and a High Powered Motor from Pololu. These three motors represent a good range of optimal size/weight, torque, and position precision. The Vex 3-Wire Servo has less than half the stall torque of the other two options and performance problems based on the group's experience, but would have more precise position control. The Micro Servo has an adequate amount of torque, but is much smaller and uses plastic gears, which is concerning if the motor encounters any rotational resistance. The Pololu High Powered Servo has a good combination of durability with metal gears high torque with a reasonable size. Since the servo is going to be run off of 5 V, the power supplies will be discussed further in the next chapter, the numbers seen in Table 2 are approximated based on data sheets.

Name	Voltage Requirements (V)	Stall Torque (Nm)	Size (mm)
Vex 3-Wire Servo	4.4 - 9.1	0.7344	23 x 14 x 25
Micro Servo	4.8 - 6.0	1.471	22.6 x 11.4 x 23.9
High Powered Servo	4.8 - 6.0	1.520	40.7 x 20.5 x 39.5

Table 2: Servo Motors

3.4 Electrical Design

The electrical systems present in the project will need to be separated into two major categories: The drone system, and the ground station. The *ground station* is what the user will be directly using to control the drone. This will also include having the capability to see any transmitted data, either from the flight controller or sensors.

The drone will house the deployment system for the FireIce; a live video feed camera, sensors, and radio systems for data transmission necessary for communication with the ground station. The drone also will have a system for power distribution to the different systems on board from the main battery.

Figure 13 shows a block diagram of the components that make up the electrical design of the project. Below we detail the different components that comprise the electrical design.

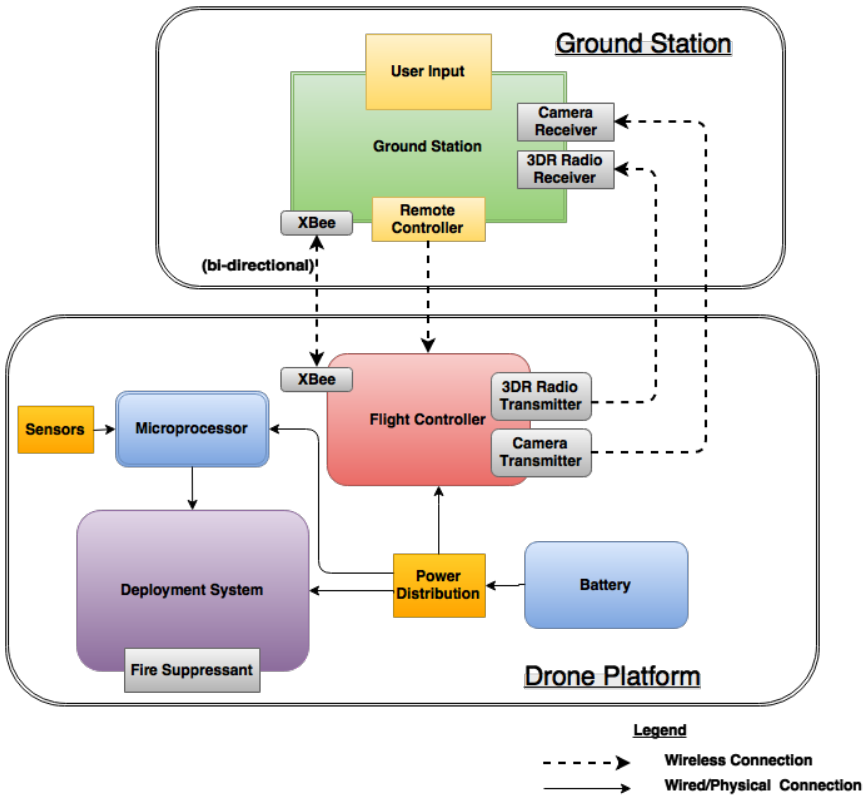

Figure 13: Electrical System Block Diagram

Drone Power Distribution

On the multi-rotor drone there are a variety of different systems that require different operating voltages and amperage. With the drone we purchased a main battery with a capacity of 22.1V at 16000mAh to power the drone frame. A larger capacity battery would weigh more, reducing the amount of payload we could have present on the drone. On the drone there would need to be a way to power the different systems of the drone. Table 3 below shows the different systems present on the drone and their operating voltage and current levels.

System	Voltage (V)	Amperage (A)
3DR-Camera	12±10%	0.05
3DR-Transmitter	7-12	0.850
3DR-Receiver	7-12	0.150
Pixhawk	5	0.500
Pressurizing System	12	2-5
Arduino	5-12	0.05
Servo for Nozzle System	12	5
Multi-rotor Drone (S1000)	22.2	40

Table 3: Power Requirements

From our analysis of the different operating points of the different systems, we decided that we would need to have two voltage regulation stages: a 12V and 5V line. We felt this would work best with our system considering the multiple systems that need a 12V supply. The pressurizing system would have the most current draw when pressurizing and deploying the FireIce.

For the 5V line, there will be another conversion stage that regulates the battery voltage to 5V at 3A. Considering the systems shown in Table 3, the voltage and amperage would be able to adequately provide enough power to the different systems.

Data Transmission

The drone system will require multiple means of communication links in order to send information down to the ground station. The need of multiple communication links stems from the different systems that will be present in the project. There will need to be a dedicated communication link for the operation of the drone. Other communication links will include the transmission of sensor data, and camera video feed. Separating each line of communication reduces the complexity of the system.

There are a variety of operating frequencies used on the system for these communication links. These systems will include: the radio to control the drone frame, a radio set for transmitting the Pixhawk flight controller information, the transmitter for the video feed camera system, and a XBee link for user control and sensor data. Table 4 breaks down the operating frequencies of the mentioned systems. Figure 14 below shows the communications architecture for the communication links present within the project.

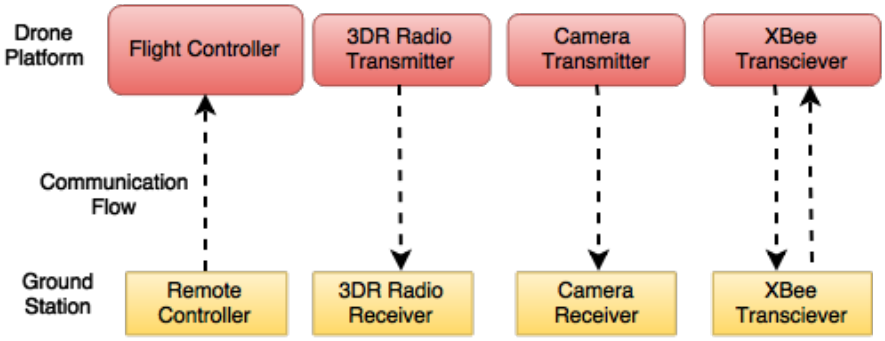

Figure 14: Communications Architecture

System	Operating Frequency
3DR-Camera	5.8GHz, 200mW
Remote Controller	2.4GHz
3DR Radio Set	915MHz
XBee Transceiver	2.4GHz

Table 4: System Operating Frequencies

The camera system is necessary to get video feed from the drone. With the camera system we intend on using the camera for flight control as well aiming for the FireIce deployment system. The camera will have an On Screen Display (OSD) module that will overlay telemetry data from the Pixhawk. This gives the operator the added benefit of knowing the orientation of the drone as well as other data from the flight controller. This other data can include the position of the drone as well as the speed.

To view the video stream at the ground station, the camera system has a dedicated transmitter and receiver that would be placed at the ground station. The operating frequency of the system is 5.8GHz. To interpret the video feed data, the receiver of the camera set has an Audio/Video (A/V) output that we plan on hooking up to a LCD monitor.

The remote controller is what will be used to interface with the Pixhawk and drone frame to fly. The current controller operates at 2.4 GHz. With the receiver end of the radio attached to the drone, there are 8-channels used to interface with the onboard systems. The first five channels are dedicated to the Pixhawk for proper control of the drone frame. The remaining three could be used as control signals for the other systems present on the drone frame.

The other operating frequency of 915MHz is dedicated to the 3DR Radio Set. The 3DR Radio Set acts as a bidirectional data link between the ground station and the Pixhawk. Similar to the OSD of the camera system, the 3DR will transmit the data from the telemetry of the Pixhawk. Using this set up will allow us another means of getting flight data. The use of this communication approach comes with Mission Planner, a dedicated software for displaying the flight data The radio set will also allow us to control the drone via a laptop as the transceiver for the ground station is connected via USB.

Deployment System

The deployment system will involve the operation of a few electro-mechanical systems such as a pressurizing system and a servo to control the direction of the nozzle. Both the pressurizing system and servo motor will need a 12V supply and between 1.5A to 5A to operate at minimum and maximum loads. To give these systems their needed power supply, a conversion stage will be implemented to drop down the supply of the main 22.2V battery of the drone into the specified operating range for the pressurizing system and servomotor. A relay will be used to control when the pressurizing system and servo motor is turned on. This relay could be controlled with a PWM driven signal from the radio receiver connected to the Pixhawk controller. The remote controller at the ground station could then have a dedicated switch to turn the PWM signal of the Pixhawk on and off. Alternatively, we could send the PWM signal from the Arduino. The user through the use of wireless communication and a GUI could send this signal.

The servo motor will be used to drive the operation of the nozzle control system of the deployment system, which will control the angle of the spray of the fire suppressant. Attached to the nozzle container in parallel with the nozzle will be the 3DR camera. This camera will have a dedicated power supply from a battery providing the 11.1V and 50mA. The camera will have a dedicated transmitter to send the video feed to the ground station via a corresponding receiver. The video feedback will provide location of the fire and the direction of the fire suppressant spray.

Sensor Integration

A number of sensors and systems will be used to provide important data to the ground station vital to the safe and efficient operation of the drone. Some of this information is obtained from the Pixhawk and drone monitoring systems already in place while other sensors and systems are added peripherals.

Flame Distance

An important piece of information is to determine our distance we are operating from the flame when we are ready to deploy our fire suppressant. One way we could accomplish this requirement is to have an IR range finder sensor setup as an added peripheral oriented to find the straight-line distance from the drone to the fire. An issue with this setup is that fire can produce IR interference, which can potentially compromise the accuracy of the sensor data.

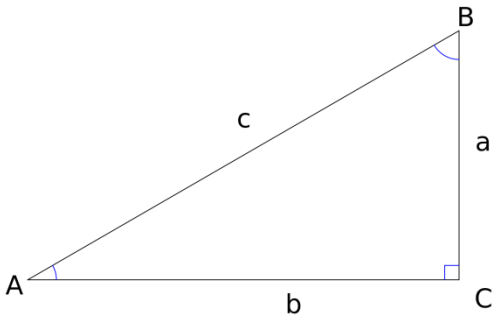

Figure 15: Law of Sines Triangle

Another way to determine our distance from the fire would be to know the height of the drone and the angle of the nozzle pointed at the fire. The height of the drone could be found using the altimeter in the Pixhawk or the GPS module of the Pixhawk, which will give us the distance of the drone normal to the ground. The angle of the nozzle could be found through using an angle sensor such as a potentiometer attached to the shaft of the nozzle-spraying container or by using the inputs of the servo used to control the nozzle's aim. Applying geometry and the Law of Sines, the distance of our drone from the fire can be determined. In Figure 15, point A is the location of the fire, point B is the location of the drone, and point

C is the ground directly below the drone. The distance ‘a’ is retrieved by the altimeter or GPS module and the angle ‘B’ is retrieved from the angle sensor or the input of the servo. Utilizing the Law of Sines, the straight distance from the drone to the fire, ‘c’, is found by the equation:

$$C = \frac{a}{sin(90 - B)}$$

Equation 3: Distance to Drone

The ground distance, ‘b’, is found by the equation:

$$b = sin(B) * c$$

Equation 4: Ground Distance

With this information, we can determine if we are in a position that is safe from overheating from the radiation of the fire as well as knowing if we are in range to spray the fire suppressant.

Camera Parallax

An issue that occurs when using a camera to direct the spray is Parallax would result from the offset angles between the camera and nozzle. Parallax is when the position of an object differs depending on where you try to observe the object. If two objects were directly in line with each other along with the spray system, the spray system would be blocked from affecting the further object. However, due to the camera being offset slightly from the nozzle, the camera could see the second object, as the line between the camera and the second object does not pass through the first object.

In Figure 16, assume Viewpoint A is the nozzle, and Viewpoint B is the camera. When trying to spray the red block, you would not be able to see the block with the camera. When attempting to spray at the blue block, the camera would be able to see the block; however, the spray would not be directed at the blue block. This effect can be minimized by angling the two lines so that rather than crossing at an arbitrary point, they cross at a specified focal point. The focal point in this instance would be the desired optimal range. To calculate the angle to tilt each object for a given focal length, you use the following equation:

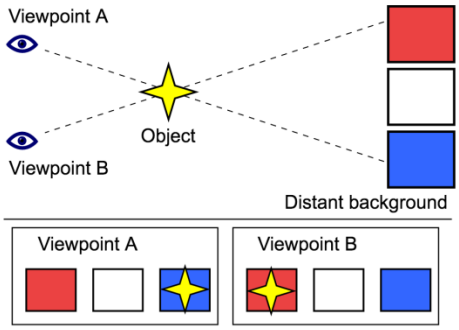

Figure 16: Parallax Example

$$\theta = cos^{-1}\left(\frac{offset\ between\ systems}{2 * focal\ length}\right)$$

Equation 5: Parallax Angle

Were as the angle that each system needs to be tilted towards the centerline. A table of calculated angles at a variety of ranges can be found in Appendix B. If the distance between the two systems were to be specified at four inches and the desired focal length were ten feet, the angle of tilt for each system would be less than a degree.

Temperature Sensor Array

As our drone and system will be operating near a fire, our system will need to know the heat directed at the drone to avoid experiencing temperatures greater than 40°C. To accomplish this requirement, a number of temperature sensors will be setup across the surface of the drone for full coverage of the systems. The sensors will send an analog voltage to the signal-processing unit, which will read the highest value, as that is the most important value. If the temperature is read as higher than 35°C then a warning will be broadcasted to the user at the ground station that the system is close to overheating.

3.5 Software Architecture

The software needs of the system can also be seen when examining the block diagram explored earlier in the block diagram in Figure 14. The places that would require software are the microprocessor, the flight controller, the remote controller, the camera receiver, the 3DR radio receiver, and the user input. Of these, off the shelf software can immediately be used for all of these sections other than the microprocessor and user input.

The software for the system needs to fulfill three major requirements. First, it needs to handle all of the in air control. Second, it needs to create a usable Graphical User Interface to allow a user on the ground to control the subsystem. Finally, it needs to manage the communication between these two systems. The air and ground systems will be shaped by what form of communication is implemented, so the first software feature investigated will be the communication structure. In addition to these functional requirements, the software must be implemented in a way that keeps the overall drone system as safe as possible.

Data Transmission

An integral portion of the project is how to control the spray system while the drone is flying. There were a number of different methods of communication considered for how to manage the data flow between the ground station and the drone. The main goals for communication were to find a reliable method of data transfer that is also safe in its implementation, and easily replicable onto additional platforms. The team investigated a number of different potential methods for communication between the ground platform and the drone in operation. The results of this investigation are detailed in the sections below.

External Transceivers

The first method considered was a pair of 2.4GHz transceivers with a range of approximately 100m. The smallest and most easily accessible option for these transceivers is a pair of XBee series two ZigBee mesh modules. This would allow for easy communication between the microcontroller running the spray system and a control laptop, however this would require mounting an additional transceiver on the drone to run the communication through. This option was initially disregarded as external transceivers were considered to add more bandwidth than is necessary to the system. External transceivers also require adding an additional data transmission frequency to the drone, causing additional data frequency interference to the area near the drone. The advantage of using this method for communication is external transceivers allows for a separate communication system from the drone remote, allowing a second person that was not the pilot to control the system. With external transceivers, the communication architecture would consist of a serial transmission line. The serial line consists of plain text communication between the connected modules.

ROS Communication

Another method that we explored was to use the Pixhawk as a Robot Operating System (ROS) node, something that is theoretically possible to do by linking the Pixhawk to a Raspberry Pi control board, however, this method would have provided far more processing power than actually needed for this application. Additionally, causing the Pixhawk to publish ROS messages would require changes to the Pixhawk firmware, adding any potential disadvantages explored in the following Pixhawk Firmware subsection, without adding any additional benefits.

Mavlink Interception

The third method investigated was to intercept the messages used by the video on OSD in order to replace some of the unnecessary data in that system with the data to be used in controlling the drone. Communication between the Pixhawk and ground systems uses a format known as mavlink to store and transmit the data from the Pixhawk sensors to the ground controller. With this system, an Arduino would be inserted between the telemetry port and the OSD board. The Arduino would receive the telemetry data, break down the mavlink message, and build a new mavlink message in the same format, except inserting our data to transmit into the message. This new mavlink message would then be transmitted to the OSD board for display on our camera.

Pixhawk Firmware

The fourth method would be to use the ADCs on the Pixhawk to send the data to the Pixhawk, and then edit the firmware loaded onto the Pixhawk in order to put the data into the appropriate location in the mavlink message before the message is sent. With this, the physical setup would not need to be changed for different groups; the only difference would be what firmware was flashed onto the Pixhawk. One downside to re-flashing the firmware is the drone would need to be re-tuned each time the Pixhawk was re-flashed. Additionally, as this would be affecting the firmware, any accidental bugs in the code could cause the drone to malfunction while flying, causing potentially dangerous situations. With rigorous testing however, this safety concern could be mitigated. The largest downside of this method is that, because when all of the data from our system is combined with the Pixhawk data the control of our system would be bundled with the control of the drone. This would force the pilot to control both the spraying system and the drone; however, the co-pilot could still act as a spotter by using the camera feed. Another problem that this method has is that any slight bug in the code could cause the drone to crash since our code would be directly affecting the drone control code.

System Control

There were two distinct methods considered for how to control the spraying system. The first was to use a separate communication method from the drone itself, allowing the spray system to be controlled by a laptop attached to some form of transceiver. This first method is best implemented if the separate communication method is also being used as the downlink to receive data being sent by the subsystem. The second method to be considered was using the three extra channels of the drone controller to send PWM signals to an Arduino, which would process the incoming signals and convert them to the appropriate signals to send to the various parts of the system. The output of these channels would be used to control the rotation of the spray system, whether or not the valve at the nozzle was open, and whether

or not the pump is running. This version of control is best used when there are not additional lines of communication being added to the full system, and instead already existing, but unused lines need to be used.

System Control Interface

To fully control the system, some form of user interface was required. If the control were to be through the physical controller, this interface would simply be the buttons used on the radio controller. The user interface becomes much more involved if a secondary software method is implemented, as the control system would need to be run through a computer, necessitating the design and implementation of a Graphical User Interface (GUI) to handle the control of the subsystem. The GUI would require features to display the temperature sensor data, as well as buttons to control the rotation of the nozzle, and to toggle the spraying. This involves interfacing with the chosen method of communication, both receiving and transmitting. This GUI could also contain an emergency stop button, in order to disable all subsystem functionality immediately in case something goes wrong inadvertently during flight.

Temperature Sensor Reading

Any communication method chosen also required the ability to read from the chosen temperature measurement system. Without this ability, the drone operator would have no way to know if they are operating the drone in unsafe situations. Due to the cost of the full system including drone, safe operation must be the highest priority for the system. There are two potential methods for reading the temperature sensors. The first is to use some form of microprocessor such as an Arduino to read the data on one of its analog in ports. This method is best used when the system will already have a microprocessor for handling interaction anyways. The second method available is to read the temperature sensors using one of the Pixhawk's available analog to digital converter (ADC) ports. ADCs are used to read an analog signal when the processor is always expecting to be handling digital data. The method of reading the temperature sensor should be chosen based on the other portions of the system, neither way has a distinct advantage over the other.

4.0 Final Design

4.1 Mechanical Final Design

Spraying System

The nozzle system has five basic parts: Nozzle Adapter, Nozzle Housing, Camera Mounting Plate, Servo Motor, and System Attachment Plate. The 0° nozzle that we chose had a plastic casing surrounding the nozzle, which we removed to replace it with our 3D printed nozzle adapter that would allow us to mount the nozzle into the nozzle housing. The nozzle adapter, which was designed to form-fit to the indents on the nozzle, provides an easy way to mount the nozzle to the housing. The nozzle housing is a 3D printed part that's purpose is to mount the nozzle to the axle that will rotate in order to give the nozzle the desired spraying motion. On the other end of the axle is the camera mounting plate. This plate allows the camera to remain in constant alignment with the nozzle. The servomotor, giving our system the spraying motion, will control the housing. The last part is the system attachment plate, which is where the whole mechanical system is mounted. This plate is the connector between each part of the mechanical system and the drone. In the following sections, each of the parts of the mechanical system will be discussed further, an exploded view of the Solidworks model of the system can be seen below in Figure 19.

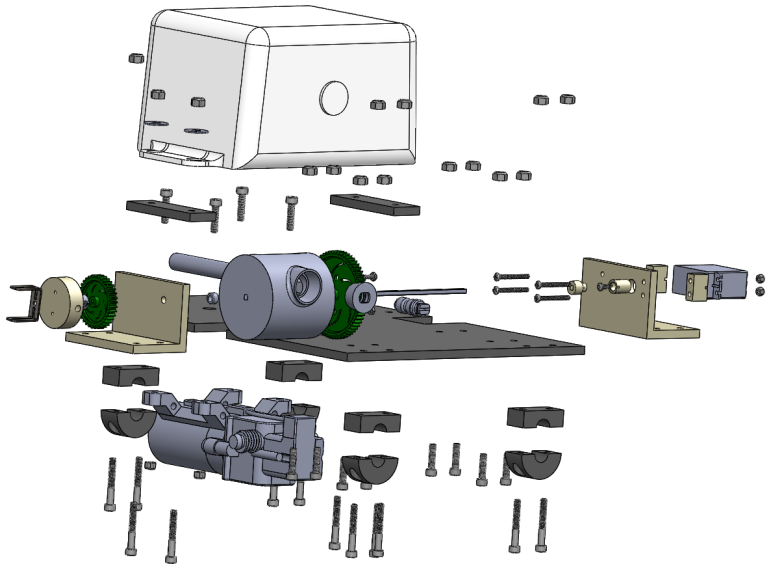

Figure 17: Full System Exploded View

Nozzle Adapter

The nozzle adapter, as seen in Figure 20, was designed to be form fitting to the nozzle on the inside and smooth on the outside in order to give an easily mountable surface. The final design was a part that would be assembled with a second one of the same part in order to form a complete seal around nozzle. Earlier iterations of the part were single pieces, however, the 3D printer material proved to be too rigid to allow the nozzle to fit in the part. A two-

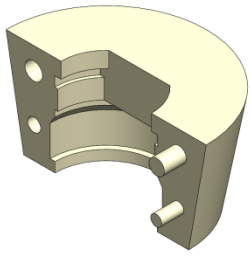

Figure 18: Nozzle Adapter

piece design allows for the part to form fit to the nozzle and provide a smooth surface to mount to the Nozzle Housing part.

Nozzle Housing

The Nozzle Housing final design has an off center nozzle placement, square hole, and 3D printed part. A picture of the final part can be seen in Figure 19. The hole where the Nozzle Adapter fits was originally in the center of the part to provide even movement of the nozzle through the rotation. After including the tubing in the design, the tubing interfered with the axle. The hole was then offset from the center to prevent the interference.

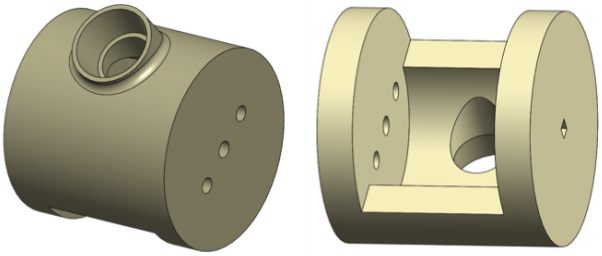

Figure 19: Nozzle Housing

Servo Motor

We considered three different servomotors: the Vex 3-Wire, a Micro Motor from Hitech, and a High Powered Motor from Pololu. The vex motor has less than half the stall torque of the other two options and performance problems based on the group's experience. The Micro Servo has a greater amount of torque than the vex motor, but is much smaller and uses plastic gears which is concerning if the motor encounters any rotational resistance. The Pololu High Powered Servo has the best combination of durability, uses metal gears, high torque and a reasonable size. Each servomotor has specifications at different voltages; however, the specifications are all around 5V. Because the specifications are not for exactly 5V the numbers in Table 5 are approximations. A picture of the Pololu High Powered Servo can be found in Figure 20.

Figure 20: High Powered Pololu Motor

Name	Voltage Requirements (V)	Stall Torque (Nm)	Size (mm)
Vex 3-Wire Servo	4.4 - 9.1	0.7344	23 x 14 x 25
Micro Servo	4.8 - 6.0	1.471	22.6 x 11.4 x 23.9
Pololu High Powered Servo	4.8 - 6.0	1.520	40.7 x 20.5 x 39.5

Table 5: Servomotor Selection

Camera Mounting Plate

The Camera Mounting Plate can be seen in Figure 21. The part is 3D printed and mounted to a vex gear to provide a stable mounting point onto the axle. The part also has a hole to fit a shaft collar to secure the axle to keep the whole assembly together. The 3DR Camera we are using has a mounting bracket that mounts to holes in the part; this provides the camera a steady way to mount to our system.

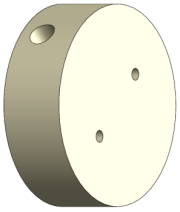

Figure 21: Camera Mounting Plate

Since the camera is securely mounted to the axle and the system, the camera can utilize the parallax calculations, explained in the Sensor Integration Section, to accurately show the image from the nozzle's perspective.

System Attachment Plate

The system attachment plate is a simple design that is dictated by the placement of the other parts of the mechanical system. To consolidate space on the plate, the diaphragm pump was moved from under the plate to on top behind the battery. This allowed the FireIce tank to be mounted to the octocopter directly without endangering the center of gravity. The plate with the parts attached to it can be found in Figure 22.

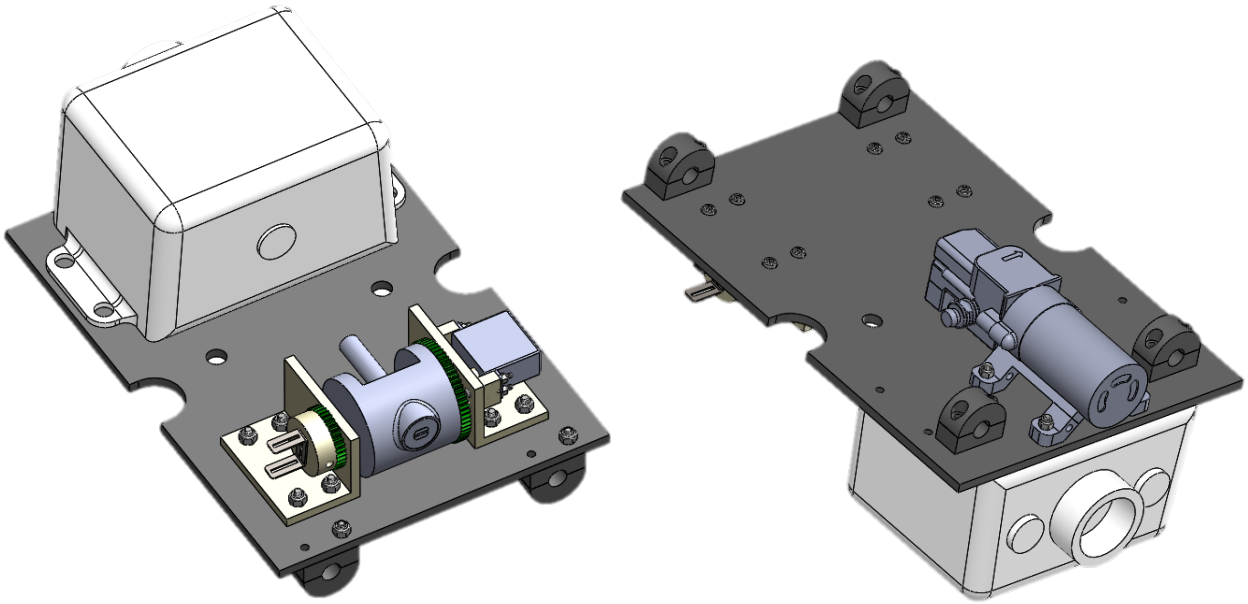

Figure 22: System Attachment Plate

4.2 Electrical Final Design

With the design of the system for the drone, the team worked to achieve the electrical design requirements specified in 3.3 Electrical Design. Specifically, we worked to address the following:

- Power distribution for the system
- Control for the wireless communication
- Control for the pump, and temperature sensor integration

The culmination of each of these systems was the construction of a PCB (Printed Circuit Board) to house each electrical component of the overall system.

We also focused on the ground station of the system and how we planned to address ground to air communication. Most of the communication for the system would be dealt by the pre made communication systems for the FPV system and flight controller.

Below we will discuss the different pieces of the electrical system. We will break it down with the different stages of the PCB and how they address the different design requirements of the system. We will also discuss how user data transmission is dealt within the system.

PCB Design

As stated above, a PCB was designed to house all stages for power distribution and computation for the overall system. We worked to include every stage of the system in one central location. The other main intent for the board was to have a rigid structure to house the electrical systems on the drone in flight.

Overall PCB Design

The PCB designed for the project is shown in Figure 23below. Below we describe each section of the PCB as well as each section's operation.

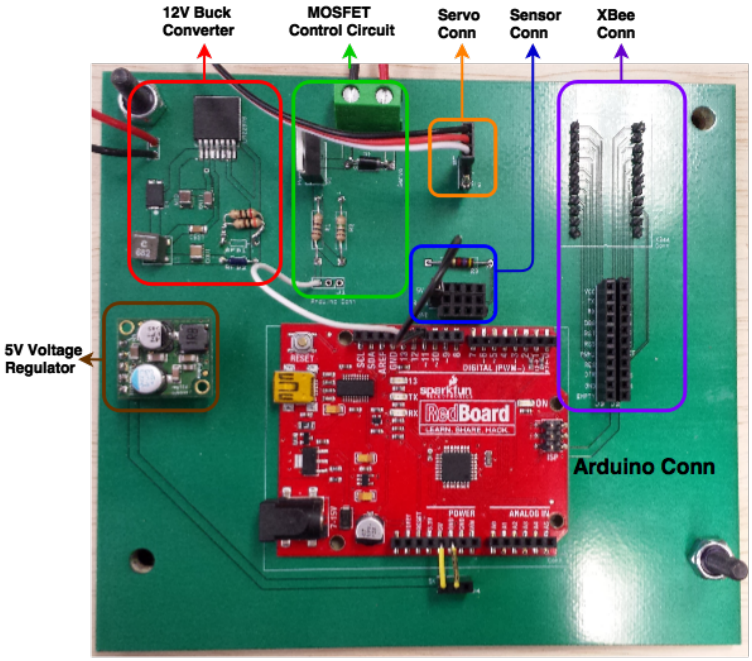

Figure 23: PCB Layout

PCB Power Distribution

For the system, we needed to have a way to provide a 12V and a 5V line to power different portions of the system. Our solution to address this was to have two buck converter circuits on the PCB. The main 22.1V battery powers both of the buck converters.

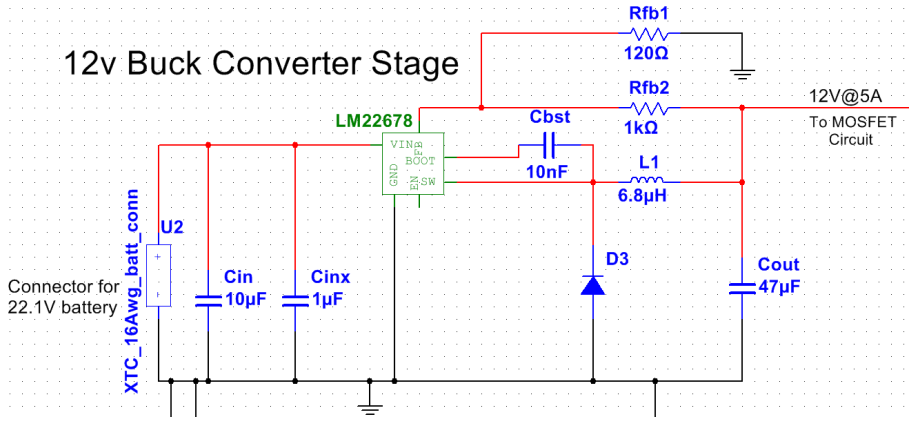

Figure 24: Buck Converter Circuit

The 12V buck converter was designed using the LM22678 step-down voltage regulator. This particular IC (Integrated Circuit) is able to regulate down a voltage upwards of 42V to a user defined voltage and amperage. Figure 24 shows the circuit used as the design for our application.

The voltage output of the buck converter circuit is dictated by the voltage divider created by the feedback resistors R1 and R2. The data sheet provided an equation to determine the appropriate resistor values to achieve the desired output. The provided equation is shown below:

$$R_{FB2} = \left[\frac{Vout}{1.285} - 1\right] * R_{FB1}$$

Equation 6: Voltage Out from Buck Converter

For R2 the recommend value was 1kΩ for the IC. With a desired value of 12V for V_{out}, we calculated the R1 to be roughly 120Ω. Using this value we were able to achieve a steady output of 12.2V. The 0.2V difference can be attributed to the imperfection in resistor values.

The voltage input/output and the inductor dictate the current output. The data sheet provides the following equation to determine the output current:

$$I_{out} = I_{CL} - \frac{(V_{in} - V_{out})}{(2 \cdot L \cdot F_{SW})} \cdot \frac{V_{out}}{V_{in}}$$

Equation 7: Current Out from Buck Converter

Where I_{CL} is the current limit with a typical value of 7.1A and F_{SW} is the switching frequency of the buck converter at 500kHz. Using an inductor of 6.8uH allowed us to have a current output max draw of 6.29A, adding an extra 1.29A to our needed amperage of at least 5A.

Through testing the pump we determined that the max current draw with FireIce would be roughly 3.7A. The current output of the buck converter circuit proved to be more than sufficient to drive the pump under

our system requirements. The safety factor of the 12V supply system was calculated to be 1.7. Safety factor was calculated using the following approach:

$$Safety\ Factor = \frac{Maximum\ Allowable\ Stress}{Working\ Stress}$$

$$Safety\ Factor = \frac{6.29A}{3.7A} = 1.7$$

Equation 8: Factor of Satety for Buck Converter

The 5V Buck Converter circuit was rated to regulate voltages up to 38V down to the desired 5V at 5A. The buck converter was purchased at Pololu. Figure 25 shows the buck converter used.

Figure 25: Pololu 5V Buck Converter

The 5V line on the PCB is used to power the Arduino Uno (microprocessor), an Xbee (wireless communication), and the Servo (nozzle painter control) of the system. Considering the potential to stall the servo, we chose the 5A voltage regulator circuit to provide sufficient amperage such that the rest of the system is not compromised. Overall, each of the voltage conversion stages provided sufficient power for each part of the circuit.

PCB Pump Control Circuit

To properly control the pump we utilized a control circuit using the IRF520 Power MOSFET (Metal Oxide Field Effect Transistor). The purpose of the circuit was to switch the power to the pump on and off via user input at the ground station. To do so the circuit takes in a control signal from an Arduino. Figure 26 shows schematic of the circuit.

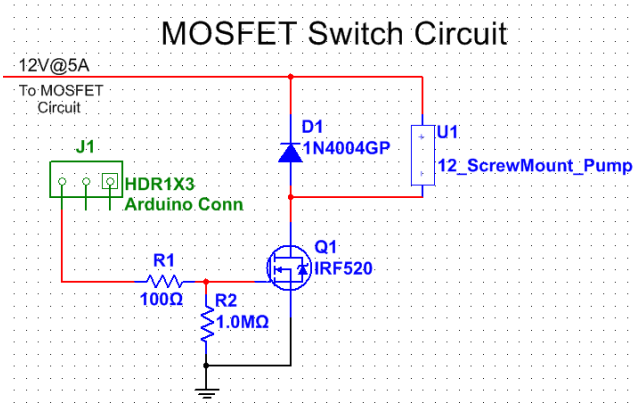

Figure 26: Pump Control Circuit

In Figure 26, the connection for the Arduino is shown as the trace going into the gate of the MOSFET from the set of header pins. The "12_ScrewMount_Pump" connector in parallel with the 1N4004GP diode shows where the pump is connected within the circuit. The 12V line from the buck converter is shown entering the cathode of the 1N4004 diode.

When the Arduino provides a positive logic signal, the MOSFET allows current to flow down across the Drain and Source of the MOSFET down to ground. As a result the pump is able to turn on and begin pumping the FireIce. When the signal from the Arduino is switched off, providing a logic low signal, the pump will also turn off. This circuit was ideal for our application as it provides the ability to wirelessly transmit a control signal from the ground station to turn on and off the pump.

PCB: Arduino and XBee Connection

On the PCB, there was a need to house the Arduino and the Xbee. To do so the components were mounted directly to the board to provide rigidity and easy access.

The Arduino Uno was mounted on the board with screws. Our rationale for mounting the Arduino in this manner was to have access to the different ports when mounted on the board.

The XBee was connected using male headers on the PCB. With the male headers the Xbee has a rigid structure to connect to allowing it to be reliably mounted on the drone. The headers then break out to a series of female headers to provide users access to the different ports of the XBee. Figure 27 shows the XBee physical connection for the PCB and Figure 28 shows the schematic connection for the XBee.

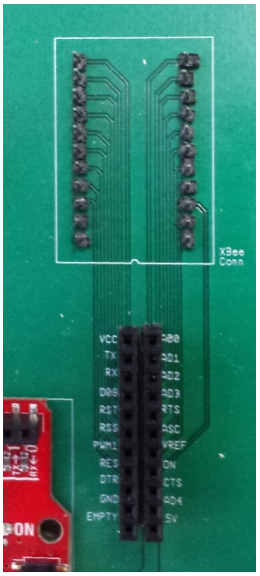

Figure 27: XBee Breakout Connection

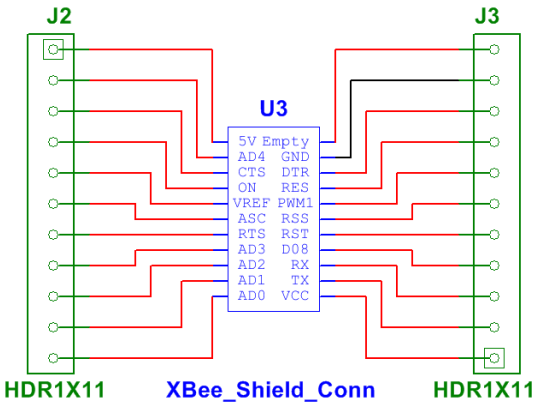

Figure 28: Schematic of XBee Breakout Connection

PCB: Sensor Integration

For the project temperature sensors are used to measure ambient temperature. We wanted to have a temperature reading to make sure the drone or the heat of the fire while in flight did not compromise our system. Our goal was to have temperature sensors distributed around the drone in order to get an average reading.

To accomplish our goal, we incorporated small cut out boards from the main PCB to be used to hold the temperature sensors. On each of the cut out boards were holes to fit zip ties through so that they can be mounted on the drone. Figure 29 shows the design for sensor cut out board.

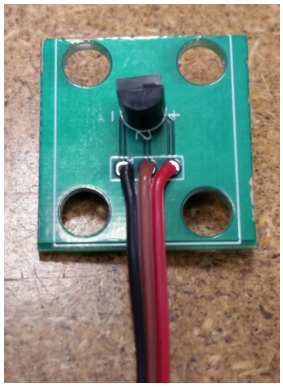

Figure 29: Temperature Sensor Board

The temperature sensors only used three connections for Ground, Signal (or data), and Power. To power and read in the temperature values we used a three-wire bus connector to a one by three male header to connect it directly to ports on the PCB.

Temperature Sensors Integration

For the drone and its subsystems, it was necessary to have temperature sensors in place to make sure our drone was not subjected to dangerous levels of heat. The temperature sensors chosen were the DS18S20 from Maxim Integrated. These temperature sensors accurately read temperatures from -55°C to 125°C with 0.5°C accuracy from -10°C to 85°C, adequately covering temperatures around our system failure temperature of 40°C.

These temperature sensors operate on a one-wire interface. Three connections were needed for the operation of the sensors: a power supply, ground, and the one wire data line. All data to and from the temperature sensor was sent along that line. A single pull-up resistor of 4.7kΩ was required to integrate the temperature sensors with our signal processing microcontroller. The voltage required to power the temperature ranged from 3.3 to 5V. A schematic of the connection and layout can be seen in Figure 30.

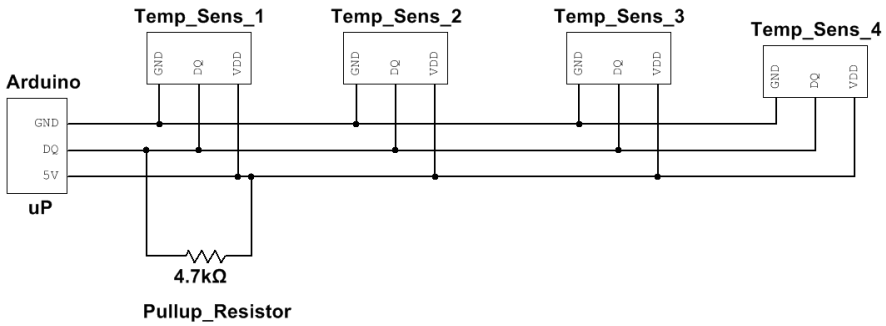

Figure 30: Temperature Sensor Connection

To have multiple DS18S20 connected to the microcontroller, they all had to be connected on the same data line, allowing only one port of the microprocessor to be occupied. By sending certain bytes to the sensors, we can activate the reading temperature and conversion functions of the DS18S20. The sensors will then in turn send bytes of data representing the temperature, which can then be converted into a decimal number of the temperature in degrees Celsius.

When sending the temperature data, the sensor also sends a unique memory address corresponding to that specific sensor. To obtain the temperature information of each individual sensor, we must first obtain the specific memory addresses of each sensor. By matching the memory address to the data being received, we can determine what temperature each sensor is outputting at that moment. This will allow us to know which area of our drone and onboard system is experiencing too much heat so we can act accordingly.

4.3 Control System Final Design

Initially, the team planned to edit the firmware on the Pixhawk to manage the air-ground communication of the system, with the extra channels on the control remote to actually control the subsystem. During development however, this was changed due to the drone safety protocols set in place by the two MQP teams using the drone. The safety protocols state that the attention of the person actively flying the drone must entirely be on the drone and not on any other systems. Because of this, the control of the subsystem needed to be separated from the control of the drone. To this end, the control method was shifted away from editing the firmware and using the extra channels on the control remote, to using a pair of XBees to maintain a separate dedicated transmission link to a ground station. This allows a separate person to control the subsystem, so the pilot does not need to worry about operating both the subsystem and the drone simultaneously.

4.4 Software Final Design

With the design of the system for the drone, the team worked to achieve the software design requirements specified in 3.5 Software Architecture. Specifically, we worked to address the following:

- User interface for easy system control
- Feedback from temperature sensors
- Communication architecture between ground and aerial platforms

In addressing these requirements, there were two pieces of software developed. One to be run on the in-air Arduino and the second to realize the implementation of the GUI. The GUI code is to be run on a laptop being used by a firefighter as the ground station for the drone. The control code is to be run on the Arduino microprocessor attached to the subsystem. Below we discuss both the implementation of the GUI and the communication structure between the GUI and the subsystem in flight.

The first piece of software is the GUI. This software is how the user interacts with the developed subsystem as a whole. Any commands for the subsystem are generated initially through the use of the GUI. The second piece of the software was the code running on the Arduino. This code handles all of the

A set of software user and epic stories was created to describe the software needs of the system. There was one overall epic story that described the full system, and then it was broken down into the user stories. The epic story reads as follows: I am a firefighter tasked with fighting a fire on a boat. I want to use a drone to fly fire suppression chemicals up to the boat and tactically spray them onto the fire to ensure maximum preservation of life. The full list of user and epic stories are detailed in Appendix E. These stories detail what exactly the system needs to be able to do when a user is operating it.

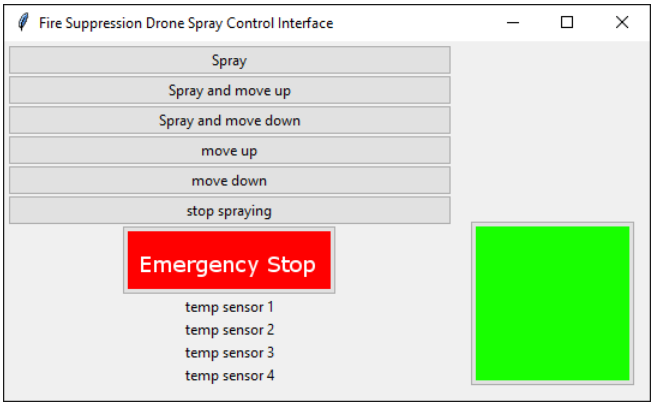

Figure 31: Ground Station GUI

Figure 31contains an image of the GUI under its final configuration. The features that were considered a requirement for the GUI were to be able to press a button to spray the FireIce, to have a button to paint FireIce up, as well as a second button to paint down. These features were combined to have buttons to do each of these individually, as well as two buttons to paint and spray at the same time. Additionally, a stop button was implemented, in order to disable the spraying immediately. The final design included button was the emergency stop button, which disables the subsystem until both the GUI and subsystem are restarted. In addition to the buttons on the GUI, a readout was added to display the temperature sensor readouts. The readout both includes a sensor by sensor display of the values, as well as a light that is green under normal circumstances, but if any of the temperature sensors go above the set warning temperature, the light will change from green to red, until all temperature sensors read under the warning level again. The GUI was programmed using Python. To manage the GUI layout, and binding, a library called TKInter was used. TKInter is the most common library used to program user interfaces in Python, it allows for the creation and layout of objects in the interface by using root panes to create windows, frames to section the windows, and then buttons and labels to add functionality to said sections. The

generation of the GUI in the code occurs between lines 196 and 296 in the code in Appendix F. To connect the GUI to the XBees, a built-in serial library that is packaged with python, called pyserial, was used to transmit the data to the transmission and receive lines of the XBee via a USB cable. The library is imported on lines 9 and 10 of the GUI code, and then is actually initialized between lines 217 and 225. The general functional structure of the GUI is shown below in a use case diagram, depicted in Figure 34.

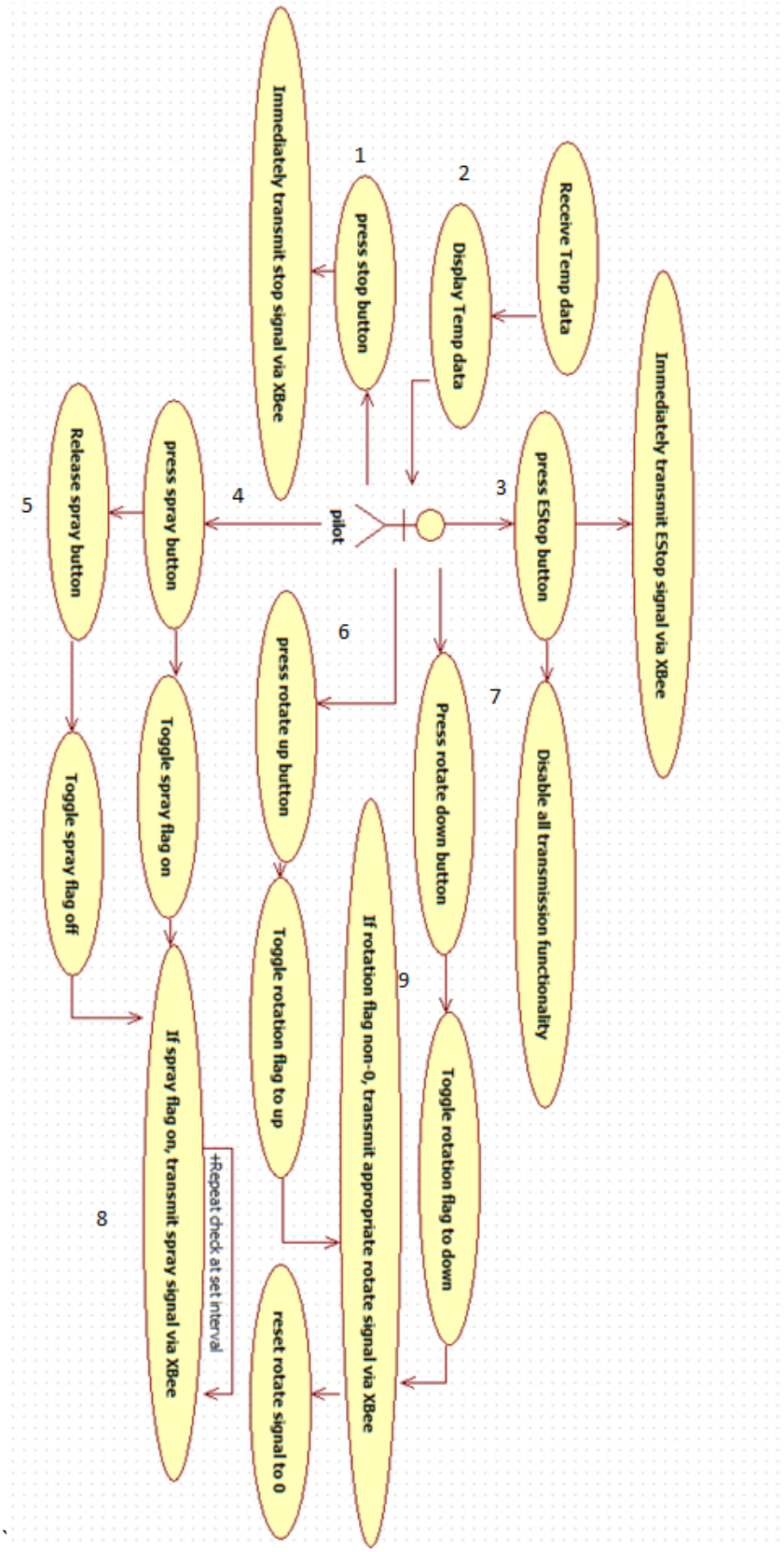

Figure 32 GUI Use Case Diagram

The numbered portions of the diagram each corresponds to a specific portion of the code. This correspondance is detailed in Table 8 below.

Table 6 GUI Use Case Code Linking

Diagram number	1	2	3	4	5	6	7	8	9
Code lines	55-58	67-154	61-64	156-158	160-163	165-167	174-176	183-194	183-194

The sections are detailed as followed:

1: This section of code explains what happens immediately when the stop button is pressed. The most important line of code here is the line `self.ser.write(b'b')`, which actually transmits the stop signal to the subsystem.

2: This section of the code is the regularly called function which reads data from the temperature sensors and then updates the GUI to show the temperatures. The exact behavior of the code here is detailed in the comments related to this code section.

3: This section of the code details exactly what happens upon the press of the emergency stop button. There are two important lines here, the first sets the local emergency stop flag to 1, preventing any further transmission, and the second transmits the E-Stop signal to the subsystem, disabling it as well.

4: This section of code is the function that is bound to pressing the spray button. Simply put, this section sets the flag controlling whether or not the system sends the spray signal to true, as long as the emergency stop flag isn't set to true.

5: This section of code is the function that is bound to releasing the either the spray or rotation buttons. It simply resets the rotation and spray flags to zero so that neither transmits.

6: This section of code is the function that is bound to pressing the rotate up button. It sets the rotation flag to 1 if the estop flag is not set.

7: This section of code is the function that is bound to pressing the rotate down button. It sets the rotation flag to -1 if the estop flag is not set.

8: This section of code is the regularly called function which checks if a spray signal should be transmitted, and then transmits it if appropriate. This function uses an if statement to check the value of the spray flag, and transmits the appropriate signal. The important portion of this segment is `self.root.after(200, self.should_send_spray)`. This line configures a delayed function call to the same function, causing the function to be called repeatedly every .2 seconds.

9 This section of code is the regularly called function which checks if a rotate signal should be transmitted, and then transmits it if appropriate. This function uses an if statement to check the values of the rotate flags, and transmits the appropriate signals. The important portion of this segment is self.root.after(200, self.should_send_spray). This line configures a delayed function call to the same function, causing the function to be called repeatedly every .2 seconds.

In addition to the GUI, software was also designed and written for the onboard Arduino microcontroller. The functionality of the in air controller is detailed in the use case diagram depicted in Figure 35 below.

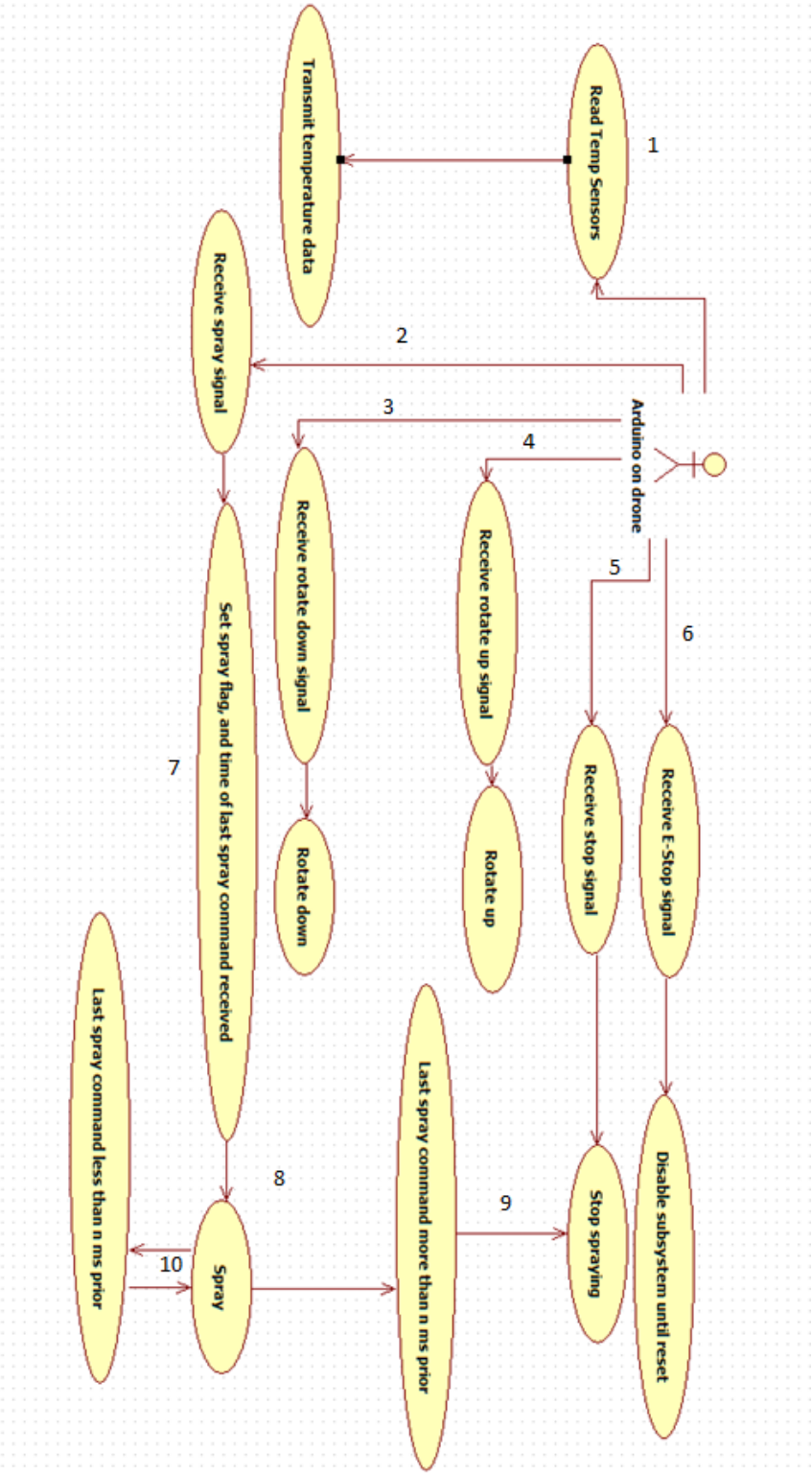

Figure 33 Drone Control Use Case Diagram

The use case diagram was again numbered for linking to specific portions of the code. These are detailed in Table 7 below.

Table 7 Drone Use Case Code Linking

Diagram number	1	2	3	4	5	6	7	8	9	10
Code Lines	141-208	78	100	95	83-88	89-94	79-80	107-109	111-114	107-110

The sections are detailed as followed:

1: This section of code is the final portion of the loop which transmits all of the temperature data, as well as the helper function that reads the temperature sensors, and properly formats each piece of temperature data for transmission.

2: This section of code checks received data to see if it is the signal meant for spraying. It uses a simple if statement to compare the received character to the desired character's decimal representation.

3: This section of code checks received data to see if it is the signal that means the system should rotate down. It uses a simple if statement to compare the received character to the desired character's decimal representation.

4: This section of code checks received data to see if it is the signal that means the system should rotate up. It uses a simple if statement to compare the received character to the desired character's decimal representation.

5: This section of code checks received data to see if it is the signal that means the system should immediately stop spraying, and does so if appropriate. It uses a simple if statement to compare the received character to the desired character's decimal representation. It then immediately resets all of the spray flags, as rotation can't be stopped mid process, and as well as sends the signal to the pump to stop spraying immediately.

6: This section of code checks received data to see if it is the signal that means the system should immediately E-Stop, and does so if appropriate. It uses a simple if statement to compare the received character to the desired character's decimal representation. It then immediately resets all of the spray flags, as rotation can't be stopped mid process, and as well as sends the signal to the pump to stop spraying immediately. In addition, it also sets the Estop flag to high, to disable the system.

7: This section of code sets the time of last spray signal received, as well as setting the spray flag to high. It is called when the spray signal is transmitted. It first sets the spraying flag to high, and then stores the current time in milliseconds for comparison to control the system spray timeout.

8: This section of code sprays if the flag is set, and the estop is not triggered, and the time of last signal was recent enough. It checks for the proper set of flags and timing and then transmits if they are all correct.

9: This section of code stops the spraying if the last spray command was not recent enough. If the time was too long ago, the system transmits a stop command to the pump to cease spraying.

10: This section of code is called each time to check if the most recent spray command occurred recently enough to keep spraying. If the time was set recently enough, the system transmits the correct spray command to the system, rather than the stop command.

The temperature sensors were read using a set of Arduino libraries provided by Dallas Temperature. These libraries read in the temperature sensors one at a time by memory address, returning the temperature in degrees celcius, or -127.00 if the sensor was read incorrectly. The transmission of this data is handled between lines 141 and 208 within the code. To transmit to the GUI, the temperature sensors were padded into a string with each temperature string being 7 characters long. This padding occurs between lines 192 and 201 within the Arduino code. For example, a set of 20 degree sensors would read as follows during transmission. 0020.000020.000020.000020.00

As the final chosen method for communication was to use a pair of transceivers to communicate the data, a method for actually formatting the data between the two systems needed to be tetermine. As such, it was decided that when transmitting data from the system to the groundstation, it was to be transmitted as plain text. When the control commands were transmitted from the groundstation to the subsystem, a set of lowercase characters were used to represent the commands. During transmission, the messages are formatted and interpreted as decimal ASCII strings. The If a command is missed, than it will likely be retransmitted shortly due to repeated transmission. If a command were to be received in error, the system would treat it as any other command received. During testing of the xbees, there was no error in communication between the systems, however, this testing was never performed at what would be considered long range for the XBees. The XBees transmit the data using a standard serial data flow, with no additional protocols added by the team in addition to the Xbee default. These commands are detailed as follows. A is spray, and needs to be continually received to keep spraying with a three second timeout. B stops the spraying as soon as it is received. Next, a received W rotates the servo to the upper position and an S rotated it to the lower position. Finally, receiving an H immediately triggers the emergency stop for the system, disabling everything until the system is reset. This overall command structure is detailed in Figure 36 below.

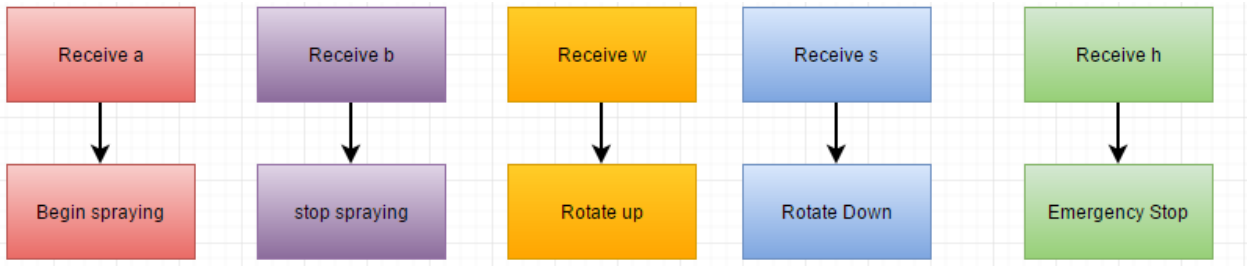

Figure 34 Command Link Structure

Flame Distance Sensor Integration

For efficiency, we want to be spraying our fire suppressant when we are in range of the fire for maximum effectiveness of our limited payload. For safety, we want to be operating at a position far enough away from fire where we can hold that position indefinitely. To acquire this data, the integration of a few sensors and systems is needed. As previously discussed in the Fire Heat Dissipation Section, the distance from the fire can be accomplished by knowing the height of the drone and the angle of which we are aimed at the fire.

To acquire the angle of our spraying appendage, we keep track of the position of the servo controlling the position of the appendage. The camera is also connected in parallel with the spraying appendage so that when the fire appears on our camera feed we know we are at the correct angle for determining the distance from the flame. The height of the drone is calculated and processed in the Pixhawk, our onboard flight controller for the drone. The Pixhawk uses a barometer and GPS signal to calculate the altitude of the drone relative to its takeoff position from the ground. This information, along with all pertinent data from the Pixhawk, is sent over telemetry to the ground station via mavlink messages. For our flame distance calculations, we will intercept a mavlink message containing the data of altitude and process it in real time with our onboard Arduino Uno microcontroller. The previous equation will then be applied using this altitude data from the Pixhawk and the position of the servo controlling the spraying appendage.

A mavlink message is a variable length series of encoded bytes relaying important data and commands to and from the flight controller to the ground control. A representation of the Mavlink message and the bytes that make up the message is shown in Figure 37[3] below

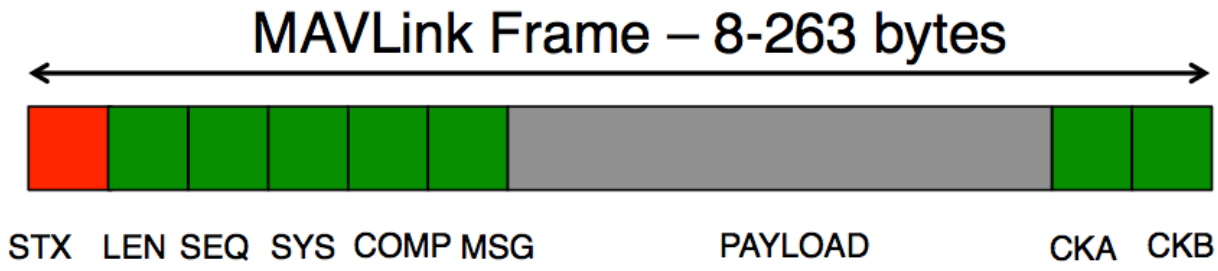

Figure 37: Mavlink Message Diagram

The order of bytes from Most Significant Byte (MSB) to Least Significant Byte (LSB) is as follows: 6 header bytes, the variable length payload bytes, and a 2-byte checksum. The 6 header bytes contain in order: a constant first byte of HxFE, byte STX, denoting the start of the message and is also related to the max size of a payload. The second byte, LEN, denotes the size of the payload of the message. The third byte, SEQ, is the sequence number of the message. The fourth and fifth bytes are the system, SYS, and component ID, COMP, respectively, and the sixth byte, MSG, is the message header. The next several bytes are the variable length payload. The size and contents of this payload depends on the specific message, which is described by the message header. The last two bytes of a mavlink message are the checksum, CKA and CKB, which does error calculations and accuracy validation of the mavlink message itself. To obtain the altitude data from the Pixhawk, we must connect one of the telemetry ports of the Pixhawk to the serial read of the Arduino. Then as we start streaming in mavlink messages we must check their message ID to see if they are the right message containing the altitude data. When the correct message has been found, we must then splice the bytes containing the altitude data, convert the bits to the actual height, and then store the value. With this value being constantly stored and updated as well as the position of the servo, we can apply the appropriate formula and obtain the approximate position of our drone from the fire.

[3] Image citation. Qgroundcontrol, “Mavlink Packet” http://qgroundcontrol.org/_media/mavlink/mavlink-packet.png?cache=

When implementing this feature, issues were encountered that kept mavlink messages from giving us the altitude of the drone. First, when utilizing Arduino mavlink libraries to aid in extracting the altitude data, the library would use up all of the RAM of the Arduino Uno creating a run time error. As the Arduino Uno has no feature for displaying how much RAM is being used, this was a difficult issue to identify. To cut down on the total amount of RAM the mavlink library was using, we changed the number of communication buffers used in the library from four to one. However, this still did not let us display the altitude properly.

Furthermore, we attempted to implement our own algorithm for decoding the altitude through a mavlink message. The algorithm would detect the STX bit and if the message header byte, MSG, was the appropriate message containing altitude then the variable payload would be stored with the size of the stored array being determined by the LEN byte. We could then convert the bytes of data corresponding to the altitude in the payload to decimal integers to get our altitude in meters. However, we were never able to implement this algorithm as when completing initial testing the only message being received was the device's heartbeat message. The heartbeat message is the message that determines whether the device is connected.

When this algorithm did not work, we attempted to use the external GPS module for altitude detection. The GPS module uses NMEA 0183 protocol to send the GPS information. This information is stored in comma-separated values, called sentences, which include relevant GPS information such as longitude, latitude, and altitude. The plan was to intercept this information using the NMEA 0183 protocol for decoding the sentence containing altitude and then extract the altitude data, but due to time constraints, this could not be implemented.

5.0 Results

All different systems of the drone system proved to be operational for the intended purpose of the project. Considering the project is meant to be a proof of concept, the team was able to address our goals of making a modular remote controlled fire containment system that was able to be placed on a drone.

The full integration of the deployment system to the drone was never fully tested due to damages sustained by the drone. Previous test flights of the drone caused the drone to need repairs, leaving little time to test our deployment system. However, the team was able to fully assemble our system and attach the system to the drone. Once the system was fully assembled, it weighed a total of 4.25 lbs without any FireIce in the container. Even with 2 quarts of water, the total weight would only be about 8.25lbs, only 75% of the maximum payload of the S1000.

5.1 Deployment System

Diaphragm Pump

The diaphragm pump that we chose for the project worked better than expected. The basic specifications of the pump met our requirements, but we had to test how well the pump would work with the FireIce. As discussed in detail in Appendix D, the testing proved that the pump would be able to spray the FireIce 19ft compared to the 12ft that we needed.

The max current draw based on the specifications was 5A, but after testing the pump only drew at most 3.8A, well below the max current draw. Since the pump was not drawing near the maximum power, the drone would be able to stay in the air longer.

There was some concern that the vibrations from the pump would disrupt the drone while it was in the air. After testing the system, however, the vibration was so small that there is little chance that it affects the drone at all.

Since the FireIce is a gel, FireIce has less leaking compared to the water that was used during testing. A feature that we did not try to implement, but ended up working in the system was a preload for the system. The pumps takes about 1s to load the system, but once tubes have been loaded, the FireIce stays in the tubes and will spray as soon as the pump is run.

Nozzle Adapter

Initially, the nozzle adapter was supposed to slide onto the nozzle to prevent leakage and pressure loss. This design, however, was abandoned when the 3D printing material proved too rigid to allow the nozzle to fit into the adapter without falling out. The complex ridges of the nozzle caused problems when trying to fit the nozzle into the adapter. This problem led to the two-piece design discussed in 4.0 Final Design. The two-piece design did not add any more leakage to the system once the adapter was fitted into the nozzle housing. The only issue with the two-piece design was that the pieces have to be glued together so the nozzle is not accessible without destroying the adapter.

Nozzle Housing

The nozzle housing had to be redesigned when the nozzle adapter was redesigned, but the redesign decreased leakage from the nozzle and mitigated the chances of the nozzle adapter falling out of the

nozzle housing. To reduce leakage, the nozzle housing was tightly fitted around the nozzle adapter so the walls of the housing kept the sides of the adapter together. The walls that held the nozzle adapter together were also tapered so that as the nozzle sprayed the reaction force would push the nozzle adapter into the housing. When the L-Brackets were redesigned and the gears removed we did not have time to redesign the housing to not require the gear, since the gear is one of the points that the axle mounts to. The axle has negligible slip on the housing when the axle is run through the whole housing since there are mounting points on both sides of the housing.

L-Brackets

The system that we designed for spraying FireIce went through many design iterations. The final design was more efficient than we expected, but the total rotation of the nozzle was smaller than we had initially anticipated. Rather than having a rotation of 0° to 90° the actual rotation was closer to 10° to 80° The decision to 3D print the brackets for the nozzle housing support was due to the weight and quality of the brackets made from the ABS Plastic. The placement of the holes was better from the 3D printed part. The whole placement was an issue with the hand-manufactured part because of human error and once the design was slightly reworked for 3D printing, the 3D printed version worked better.

Part of the redesign of the L-Brackets was removing the mounting holes and gears from the brackets. The gears were initially included in case the servo motor was not able rotate the nozzle housing; however, once the system was tested with 1:1 gear ratio the servo was able to rotate the nozzle housing. Removing the gearing aspect of the L-Brackets decreased the weight and complexity of the part making the bracket more functional.

FireIce Tubing

The tubing that we used to run between the diaphragm pump, the tank, and the nozzle was Flexible PVC tubing. The tubing worked well for initial testing of the system and had no problems handling the pressures from the diaphragm pump. The tubing did restrict the motion of the nozzle housing more than expected down to 10° to 80°, as mentioned before. The tubing started to bulge slightly once FireIce was run through the tubing; however, the tubing did not leak or break during testing.

After testing, the tubing was cut much shorter to allow for a more compact system and after many rotations of the nozzle housing the tube started hardening. The hardening of the tubing prevented the nozzle housing from rotating almost entirely. Overall, the tubing successfully fulfilled the requirements even with the drawbacks.

5.2 PCB Design

The PCB for the project proved to work as expected. The intent of the board was to fulfill the following criteria:

- Provide a reliable 12V line for the Pump system
- Provide a reliable 5V line for the Arduino, XBee, and Temperature Sensor Network
- Contain a sturdy mount for the Arduino and XBee
- Contain a switching circuit to turn on and off the Pump system

Considering that there had to be minor modifications to the PCB, the board overall was able to satisfy each of the above criteria.

Each of the individual circuit sections of the PCB proved to work reliably. The 12V buck converter circuit successfully provided a steady 12.2V at 5A. This proved to be sufficient for the pump system even when pumping FireIce as it drew roughly 3.8A. The 5V Pololu Buck Converter also worked as intended, providing sufficient power to the Arduino, XBee, and temperature sensors. The MOSFET switching circuit was able to successfully switch on and off the pump system with a control signal from the Arduino.

When developing the PCB there were many roadblocks. Many of the issues stemmed from the lack of experience in designing a PCB. There was significant research and work done learning the Ultiboard software to develop the PCB. There were some minor mistakes made when choosing through hole sizes for certain components as well as choosing the appropriate settings for solder masks on components. There were unforeseen roadblocks like the onboard connection of the 5V Buck converter. Since it was connected on the board using standard 0.1in header pins, the converter sat higher on the board then expected, compromising the rigidity of how it was mounted.

There were also some issues with properly using the XBee. The top pins on the XBee shields include a voltage regulator to step the input voltage down from the 5V in to the 3.3V used by the XBee while the pins on the bottom of the shield do not. As a result, if 5V is applied directly to the bottom of the shield, the voltage will damage both the shield and the XBee, requiring both to be replaced. We were initially unaware of this and burnt out two XBees in the process. As such, in a future redesign, in order to remove this danger, the PCB should include a linear voltage regulator between each of the breakout pins for the XBee and the ports where the shield plugs into the board.

5.3 System GUI

As a whole, the GUI functioned fully for all system testing. The largest problem still present in the code is it takes time to cycle through the code, and therefore occasionally takes additional time to respond to button presses. Other than this, the final GUI did not have any noticeable problems. As of now, all attempts to determine exactly what was causing this issue, as well as any attempts to fix the problem have been unsuccessful. To attempt to fix this problem, the transmission rate of the GUI was slowed down, as an attempt to reduce how often the GUI blocked, however this effort was unsuccessful. The likely next best method to attempt to repair this problem would be to rewrite the GUI to use multiple threads running in parallel for running the GUI and handling communication separately.

While developing the GUI, the initial method of implementation was to use a program called Python GUI builder, or PyGuBu, which allows the user to drag and drop GUI features and generates the code to run the GUI. The code that was generated by PyGuBu worked to run the GUI, however, the GUI had a bug where closing the GUI did not stop the program. The code was then rewritten from scratch without PyGuBu, which resolved this problem. The final code for the GUI is detailed in Appendix F. The GUI is layed out in the form of a root window, which contains two frames, one for all of the buttons and temperature information, and one that just contains the alert light box for displaying the warning light. From there, an image is created for each of the picture based buttons, specifically the emergency stop, and the alert light. Then, seven buttons are placed in the main frame of the GUI, one for each button, and the

relevant functions were bound to them. The Estop button also had the Estop image attached to it. After the buttons were created in the main frame, a button was added to the alert frame. This button has no functions attached to it, but it does have the safe temperature image added to it initially. After all the buttons are created, a set of four labels are placed below the control buttons. These labels are used for the display of the temperature sensor data.

The next problem the code encountered during development was the code written for the drone expects regularly received transmissions to continue spraying. With the `onClick` listeners that were initially used for implementation, the listener would transmit once when the button was pressed, rather than running continuously. This problem was fixed by creating a pair of triggers, one for when the mouse was pressed above the button, and one for when the mouse was released. The on-press trigger set a flag to on, and then the release trigger turned the flag off. A continuously running function watched this flag, and transmitted the required symbols via the XBee while the flag was on. The following code shows the binding of the on and off triggers to the button.

```
self.spray_down_button = ttk.Button(self.mainframe, text="Spray and rotate down",
command=self.on_spray_down_button_clicked, width=60)  # create button to spray and rotate down
self.spray_down_button.grid(column=2, row=2)
self.spray_down_button.bind('<ButtonPress-1>',self.spray_move_down)  # bind function to start spraying and move servo down
self.spray_down_button.bind('<ButtonRelease-1>',self.stop_spraying)  # bind function to stop spraying
```

This next section of code shows the functions that are bound to the button press and release.

```
def spray_move_down(self, event):
    if self.e_stop_flag == 0:
        self.servoFlag = -1
        self.spray_flag = 1
def stop_spraying(self, event):
    # print("stop called")
    self.spray_flag = 0
    self.servoFlag = 0
```

Finally, the following section of code shows the function that checks these flags regularly and transmits as needed.

```
def should_send_spray(self):  # A function which is called regularly in order to make
sure that the servo receives
    # its move commands and the pump regularly receives spray commands while one of
the spray commands is held
    if self.spray_flag == 1:  # if spray held, send spray command by serial
        # print("here")
        self.ser.write(b'a')

    if self.servoFlag == 1:
        self.ser.write(b'w')
    elif self.servoFlag == -1:
        self.ser.write(b's')

    self.root.after(200, self.should_send_spray)  # this function again after 200
milliseconds
```

There are 6 total functions that are used in various combinations to handle the different modes of buttons. The after call in the should_send_spray function causes the function to be called every 200 milliseconds.

The button binding and the functions bound to the buttons control which flags are set to what values in order to send the correct signals to the Arduino.

The next issue that came up was portability of the program. When initially written, the images used for the temperature warnings were referred to by their absolute path on the hard drive, rather than the path relative to the program. The other part affecting the portability was the path of the XBee when plugged into a computer. On a computer running Windows 10, the XBee's path would appear as COM3, and on a computer running Mac OSX, the XBee would appear as /dev/cu.usbserial-A603UFHF. Initially, the XBee's path was hard coded into the GUI code, however, this was later updated to use the absolute path of /dev/cu.usbserial-A603UFHF if that path appeared in the list of available communication ports, and to use the first available communication port if it appeared with a different name. The serial initialization is done using the following section of code:

```
ports = list(serial.tools.list_ports.comports())
for p in ports: #loop through all ports looking for /dev/cu.usbserial-A603UFHF
    if(p[0] == '/dev/cu.usbserial-A603UFHF'):
        self.portflag = 1
    print(p)
if self.portflag == 0:
    self.ser = serial.Serial(str(ports[0][0]), 9600)  # initialize the serial
communication
else:
    self.ser = serial.Serial('/dev/cu.usbserial-A603UFHF', 9600)  # initialize the
serial communication
```

The code starts by finding a list of all of the COM ports visible on the system. Next the program loops through the list of ports and checks each item to see if it includes the true device path of /dev/cu.usbserial-A603UFHF. If it finds the exact path, it sets a flag to 1, so that the program knows the absolute path has been found. Then, if the portflag was not set, it initializes the serial communication to the first communication device in the list of available devices. If it were set to 1, then the code initializes the communication using the true path that the code knows is available.

5.4 Subsystem Control

The code running on the Arduino receiving the control sent by the GUI, and interpreting that data into system commands worked without incident. All data received was processed and interpreted into controls. The spray timeout worked successfully, though needed to be set at three seconds, rather than the desired one second, due to an issue with transmission blocking between the XBees causing occasional one and a half to two second delays in transmission of only spray commands. The other commands appeared to process through to the Arduino as soon as they were sent.

5.5 Sensor Array

The temperature-sensing array worked well to provide accurate and reliable data in a timely manner. The DS18S20 temperature sensors used streamed data at a rate of one reading per second to the Arduino using a one-wire interface. This allowed us to simultaneously read the temperature of all of the sensors at the same time. By using the unique memory address of each sensor, we were able to tell which temperature sensors were experiencing changes. This would allow the user to know which areas were experiencing more heat and adjust accordingly. Additionally, code was implemented to signal a warning to the user if the temperature sensor read a temperature exceeding 35°C, as the electronics of our system will start

failing at 40°C. There was an issue where two of the temperature sensors stopped working properly for unknown reasons, but this issue was not replicated when replaced with new temperature sensors.

For determining the distance from the fire, the sensor integration required for this task did not function correctly. The major issue encountered was extracting the altitude data from the Pixhawk. There were issues with using an Arduino Mavlink library on the Arduino Uno because of memory limitations. Using the library would use up all of the RAM memory on the Arduino Uno and cause a runtime error. This would prevent the needed data from streaming. Attempts to limit the memory requirements of the library by reducing the number of communication buffers were met with little success as the only message being properly received was the heartbeat message. This issue of memory limitation could be easily countered by using a microcontroller with greater RAM, such as an Arduino Mega, a Rasberry Pi, or another microcontroller with RAM greater than the Uno's 2kB of RAM.

Implementing the necessary decoding algorithm for mavlink messages was met with limited success as the mavlink messages being broadcasted weren't fully being received over serial to the Arduino Uno. Again, only the heartbeat message was received. Further research into using mavlink communication could yield the reason as to why only the heartbeat message was received. From observation, it could be that a particular start data streaming message is meant to be sent to the Pixhawk to enable it to send the necessary mavlink messages over the telemetry port used by the Arduino.

When mavlink communication did not work for us, we attempted to use the external GPS module provided with the Pixhawk to determine the altitude data. The messages sent by the GPS module used the NMEA 0183 protocol. However, due to time constraints using the GPS module for extracting the altitude information was never implemented. Further research and testing with the GPS module could provide the needed altitude data.

While we were able to keep track of our servo position necessary for determining our angle of the nozzle, without also knowing the height of the drone we were not able to calculate the distance from the fire.

6.0 Conclusions

The goal of this project was to develop a prototype of a modular fire suppression system. Based on the tests that the team ran on the system with small-scale fires the system works as intended. As with any project, there were design choices that were change during implementation, but all of these changes lead to the final design that was able to extinguish a small fire and was able to attach to the drone.

Mechanically, the system was assembled with the knowledge that this was an exploratory project so not every part was manufactured with small tolerances. When assembling the system every part was mounted, but not always where might be an optimal placement since this iteration of the project was intended as a proof of concept.

Electrically, the PCB has all the needed components to run the system unless significantly different sensors or features are added. However, the PCB was not designed with space efficiency as apriority, rather the functionality was more important. Some of the circuitry of the PCB also had to be changed after the PCB was printed so if the board is going to be used again the board could be reprinted with these changes in mind.

As for the software side of the project, the main goal was to support the electrical and mechanical aspects of the system and give the user a way to control our system. The software successfully implemented control of the mechanical system and reading of the sensors. As with almost all software there is much room for improvement, there are always new features that can be added to the system through software.

As a group, we compiled different ideas on how to improve the project to make the capabilities of a firefighting drone more favorable. The recommendations can be separated into improving the already implemented system and future reach goals.

The following is a table of the requirements we set forth at the beginning of the Requirements and Initial Design section:

Requirement	Result
Drone System	DJI S1000 chosen
Effective fire extinguishing agent	FireIce chosen, highly effective fighting fire and providing a path of egress
Modular System	System is complete detachable
Sensor Signal Processing	Arduino used and successfully implemented
Easily add FireIce to container	System container has cap for easy access to FireIce
Ground Station for System communication	Telemetry data transmission, video transmission, sensor read outs, and control of the system all

	through the use of a GUI
Video feed system	Tested successfully, but never mounted onto the system
Heat protection for the drone	Research completed, but no heat shielding installed on drone
Container for FireIce	COTS container chosen holds 2quarts of liquid
Get the FireIce from the drone to the base of the fire	Diaphragm pump to pressurize the system
Nozzle needs to spray at least 12ft	0° pressure washing nozzle chosen, which sprays 19ft
Painting motion for nozzle	Nozzle housing designed and rotated by a high powered servo motor
Power from battery needs to also provide 12V and 5V	PCB designed that provides 12V and 5V
Paralax solution	Research complete, but since the camera was never implemented the angular solution was never implemented
Temperature Sensing	Temperature sensor array created and tested successfully

Table 8: Requirements and Results

7.0 Recommendations for Future Projects

7.1 Mechanical

1. Develop a complete and accurate model of the drone in Solidworks. We used an almost accurate model, however, the model does not include every part on the drone, and we did not entirely verify as accurate. The fire extinguishing system has a Solidworks model that is up to date with the current state of the system. The model that we used also was missing all of the mates with in the system.
2. Adding temperature shielding to the drone in key areas would be necessary before any large scale testing. We mentioned this in the System requirements section of the paper, however, due to time restrictions we were never able to implement this on to the drone.
3. The tubing on the drone is another area that could use further work. The tubing that we used was acceptable, but caused restrictions and required careful routing. Future teams should research a replacement for the tubing on the system. This would allow for more rotation and hopefully more efficient tubing paths.
4. The FireIce container also has a couple of aspects that needs further design. First, to decrease the amount of FireIce left in the container after the nozzle cannot spray anymore, the container should be angled so that the FireIce is directed towards the tubes. Future teams could also redesign the general mounting of the tank so that the tank is balanced better with the rest of the drone to reduce the strain on the rear motors.
5. The spraying system could also benefit from further design. To increase the efficiency even further, future teams could add another degree of rotation to allow for even more coverage from a stationary drone. The nozzle housing would be more consistent if there was some metal coating around the axle connection points to prevent the 3D printed plastic from deforming.
6. The spray pattern of the nozzle would be another area that requires further testing. The team had discussed the idea of testing spray pattern to be able to determine the most effective nozzle not just in distance but also firefighting capability. Testing would be good for every part of the system so that there are numbers to back up the fact that the system works.
7. Another long-term transition for the project would be to transition to a diesel powered drone of the same scale or larger. Diesel powered drones have a significantly higher flight time compared to that of battery powered drones. They are also able to handle heavier payloads, which in turn will allow the drone to contain more fire retardant. Companies like Top Flight already implement diesel-powered drones for agricultural spraying purposes.

7.2 Electrical

1. For the logic control of the system, it would be beneficial to transition away from the Arduino to a dedicated microprocessor chip. For the project, we ran into issues with the limitations of the Arduino, specifically memory. Having a dedicated microprocessor would allow us to run the different portions of the system more effectively.

2. The team discussed, but was not able to implement a thermal camera to the system to increase the fire locating power of the drone. This would allow the drone to rise to a high vantage point and locate fires, possibly even before first responders.
3. Having a way to determine how full the tank is would be another great addition to the project. Developing a sensor system that could detect the current level of FireIce would be a great addition to the project. This would allow the user to better manage their suppressant and allocate it to accordingly.

7.3 Software

1. As the flame distance integration of the project was not completed, it is recommended to continue the process of determining the altitude data from the Pixhawk using Mavlink messages. Alternatively, developing an accurate flame distance sensor would accomplish this goal and have further applications in fire protection beyond this project.
2. With the capabilities of the S1000 drone and the Pixhawk flight controller, integrating autonomous functionality would greatly enhance the capabilities of the project. Using the current system, we could use GPS localization to set waypoints to desired locations. This would be key for reaching a fire.
3. Future reach goals of the project delve into recent advancements in drone technology. Incorporating swarm robotics, for instance, would be a unique approach to tackle large-scale fires. Considering that the amount of fire retardant that is able to be held by our current system is only sufficient for small scale fires, multiple drones could combat larger fires.
4. Implementing a heads up display on the video feedback screen of the drone camera would be another great addition to this project. Potentially, the heads up display could accurately show where the spray would land on the screen by using information such as the drone's height, velocity, and wind speed.

Appendix A

Decision Making Process

1(Worst) -> 10(Best)	Feasibility of Application	Put out Type A fire	Put out Type B fire	Put out Type C fire	Cost (for our project)	Environmental Impact	Safe for people	Longevity of Suppression	Weight	Fire containment	Total
Salt Water pump	7							10	4	6	21
Sound	4							8	4	5	16
Dry Chemicals	6							6	7	4	19
Electrostatic/Magnetic Fields	2							8	3	3	13
Water Mist	7							7	6	6	20
CO2	6							7	7	4	20
Chemical Grenade	6							3	7	3	16
Foam	7							7	7	7	21
Fluoroform	N/A										0
Halon	N/A										0
Gel (Firelce)	7							8	8	10	23
Wetting Agents	7							6	7	7	20
Our best choices (ranked) 3 = best 1 = worst											
Firelce	2	3	3	3	3	3	3	2	3	3	25
Pump System Salt Water	1	2	1	1	2	3	3	3	1	1	17
Foam	3	1	2	3	2	1	1	1	2	2	16

Figure 38: Decision Matrix for Fire Suppression System Selection

The first phase of selection is the top portion of the decision matrix, with all considered fire suppression systems. The Bottom 4 rows contain the second phase ranked decision matrix of the top three choices selected from the first phase.

Appendix B

Distance to fire	Angle to tilt camera and spray
1	9.594068
2	4.780192
3	3.184739
4	2.388015
5	1.910213
6	1.591754
7	1.364314
8	1.193748
9	1.061094
10	0.954974
11	0.868151
12	0.7958
13	0.734581
14	0.682109
15	0.636633
16	0.596842
17	0.561732
18	0.530524
19	0.502601
20	0.47747
21	0.454733
22	0.434063
23	0.41519
24	0.397891
25	0.381975
26	0.367283
27	0.35368
28	0.341048
29	0.329288
30	0.318312
31	0.308043
32	0.298417
33	0.289374
34	0.280863

Table 9: Parallax Data

Appendix C

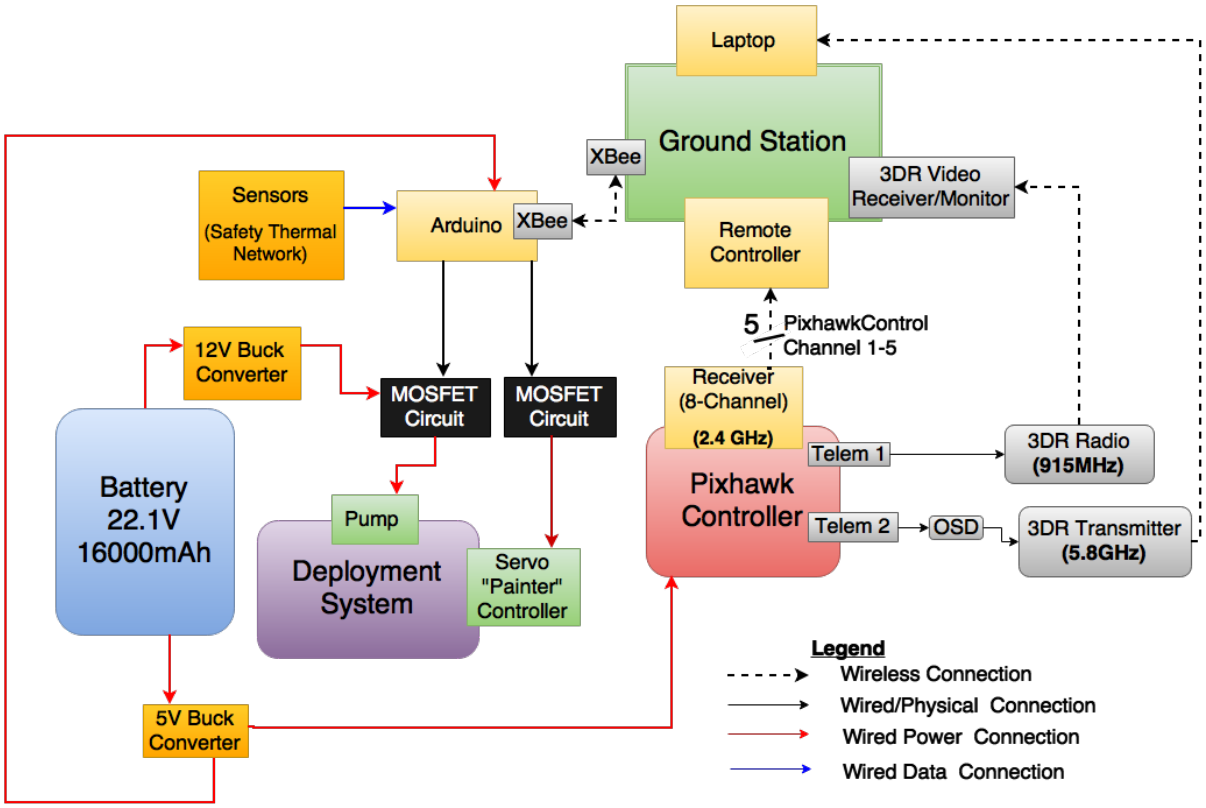

Figure 39: System Block Diagram

Appendix D

To determine the best nozzle for our application we ran tests on many different nozzles until we found the one that met all our criteria. The first test determined the spray distance of each nozzle using water. As seen in Figure 40, we measured 2ft off the ground, held the nozzle parallel with the ground and pumped water through the nozzle, and recorded how far it went.

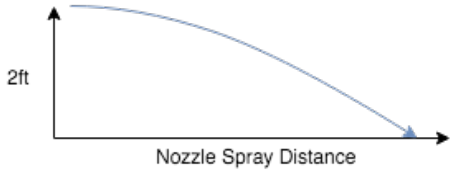

Figure 35: Nozzle Testing

We tested seven different nozzles: power washing nozzles of 0°, 15°, 25°, and 40°; a variable angle nozzle; and a nozzle for a fire extinguisher. The larger the angle on the power washing nozzles the less range the nozzles had, as seen in Table 10 at the end of this section. The best results were from the 0° and the second 15° nozzle for water with ranges of 12ft-15ft and 14ft-15ft respectively; the rest of the results can be seen in Table 10.

The distance covered by the nozzle spray was a range due to inconsistencies when spraying. There were outside factors like wind that contributed to the varied results of the nozzle testing. The worst nozzles that we tested were the 25° and 40° nozzles. They both reached a much smaller range than the other nozzles and the spray pattern began to degrade, creating a mist, towards the end of the stream.

For the next test, we ran the FireIce solution through our system. Because of the increased viscosity of FireIce, the distance of the spray increased compared to testing with water. The increase in distance can be attributed to an increase in the pressure in the tubing system. Using FireIce instead of water also helped the spray to stay consistent allowing us to take more accurate measurements.

We also measured the amperage that the pump drew while pumping the FireIce. We deduced that the increase in viscosity would cause a significant increase in amperage draw. The amperage draw did increase, but it did not reach the max current draw for the pump. For this test we only tested the nozzles that met our minimum requirements of 12ft when spraying with water, which were the power washing 0°, 15°, and the Fire Extinguisher nozzle.

The 0° nozzle provided the farthest range with a distance of 19ft. The Fire Extinguisher nozzle decreases in the distance the nozzle can spray when FireIce was run through the nozzle. Since the Fire Extinguisher nozzle had a larger outlet hole the increase in viscosity caused more internal friction when spraying. However, the larger outlet hole did not allow the pressure to build up more than before.

For the final design we decided to use the 0° nozzle because of its greater range compared to the other tested nozzles. The design of the nozzle also allowed us to remove the plastic casing and design our own housing for the nozzle to attach it to our system, this will be talked about further in the following section.

Nozzle	Distance [Water] (ft)	Distance [FireIce] (ft)	Amperage (A)
0°	12 - 15	19	3.6
15°	13-14	13	3.7
25°	4	N/A	N/A
40°	3	N/A	N/A
Open (Variable)	8.5 - 9	N/A	N/A
Closed (Variable)	8 - 9	N/A	N/A
15° (Fire Extinguisher)	14 - 15	10.5	2.8

Table 10: Nozzle Testing

Appendix E

Epic story
I am a firefighter tasked with fighting a fire on a boat. I want to use a drone to fly fire suppression chemicals up to the boat and tactically spray them onto the fire to ensure maximum preservation of life.
User stories.
I am a firefighter who wants to manually fly the drone for optimum positioning.

I am a firefighter who wants to give the drone a path to fly for specified positioning.

I am a firefighter who wants the drone to hold its position exactly after I have maneuvered it into position.

I am a firefighter who wants to know if the drone is getting too low on battery to be able to safely return and land.

I am a firefighter who wants to know if the location of the drone is too close to the fire to be able to operate safely.

I am a firefighter who wants to be able to spray the fire suppressant by pressing a button.

I am a firefighter who wants to be able to rotate the nozzle to paint with the fire suppressant.

I am a firefighter who wants to be able to rotate the nozzle while the fire suppressant is being sprayed.

I am a firefighter who wants to be able to press a button to immediately stop spraying.

I am a firefighter who wants to be able to press a button to fully disable the suppressant subsystem until the drone lands, in case of severe malfunction of the subsystem.

I am a firefighter who wants the control of the drone to be as reliable as possible to prevent potential malfunctions.

Appendix F

```
__author__ = 'David'
import sys
import os
import serial
import time
import re
from PIL import Image as im
from PIL import ImageTk
import serial
import serial.tools.list_ports

# import the proper version of tkinter, this will make sure that the correct version
for python three is imported,
# because terrible library naming
try:
    import tkinter as tk
    from tkinter import messagebox
    from tkinter import *
    from tkinter import ttk
except:
    import Tkinter as tk
    import tkMessageBox as messagebox
    from Tkinter import *
    from tkinter import ttk


class MQPGui:
    def on_up_button_clicked(self): # define callback for up button
        if(self.e_stop_flag==0):
            # self.servoFlag = 1
            self.ser.write(b'w')


    def on_down_button_clicked(self): # define callback for down button
        if(self.e_stop_flag==0):
            # self.servoFlag = -1
            self.ser.write(b's')


    def on_spray_up_button_clicked(self): # define callback for spray up button
        if(self.e_stop_flag==0):
            #self.servoFlag = 1
            self.ser.write(b'w')


    def on_spray_down_button_clicked(self):# define callback for spray down button
        if(self.e_stop_flag==0):
            #self.servoFlag = -1
            self.ser.write(b's')


    def on_spray_button_clicked(self): # define callback for spray button
        if(self.e_stop_flag==0):
            quack = 0

    def on_stop_button_clicked(self): # define callback for stop button
        self.spray_flag = 0
        #self.alertButton.config(image=self.photo3)
        self.ser.write(b'b')
```

```
    def on_e_stop_button_clicked(self): # define callback for emergency stop button
        self.spray_flag = 0
        self.e_stop_flag = 1
        self.ser.write(b'h')


    def update_temp_sensors(self): #create function for continual updating of
temperature sensors
        if(self.spray_flag == 1): #make sure this function doesn't interfere with
spraying by sending a spray command if spray is currently toggled on.
            #print("here")
            self.ser.write(b'a')
        temps = [] # set up array to read temperature sensor data into.
        temps.append(0)
        temps.append(0)
        temps.append(0)
        temps.append(0)
        # read in the temperatures
        tempsTemp = str(self.ser.readline()) #read in first set of temperature data in
buffer.
        #remove non-decimal values from the temperatures
        non_decimal = re.compile(r'[^-\d.]+') #create a regular expression to filter
non-decimal or negative signs out of the received string
        tempsTemp = non_decimal.sub('',tempsTemp) #Apply regular expression to string
        #split the temperature string into its four separate sensors.
        temps[0]=tempsTemp[0:7]
        temps[1]=tempsTemp[7:14]
        temps[2]=tempsTemp[14:21]
        temps[3]=tempsTemp[21:28]
        if(len(temps[0])>=7 and len(temps[1])>=7 and len(temps[2])>=7 and
len(temps[3])>=7): #check to see if all of the values are long enough to be properly
read.
            if temps[0][6] == "-": #if the last digit of the any temp sensors is a
negative sign, a shift occurred, so the negative needs to get moved down
                temps[1] = "-" + temps[1]
                temps[0]= temps[0][0:6]+"0"

            if temps[1][6] == "-":
                temps[2] = "-" + temps[2]
                temps[1]= temps[1][0:6]+"0"
            if temps[2][6] == "-":
                temps[3] = "-" + temps[3]
                temps[2]= temps[2][0:6]+"0"
            # Clear any leading zeros or zeros that were accidentally converted to 8s.
            if temps[0][0] ==  "0" or temps[0][0] == "8":
                if temps[0][1] == "0"or temps[0][0] == "8":
                    temps[0] = temps[0][2:]
                else:
                    temps[0] = temps[0][1:]
            if temps[1][0] ==  "0"or temps[1][0] == "8":
                if temps[1][1] == "0"or temps[1][1] == "8":
                    temps[1] = temps[1][2:]
                else:
                    temps[1] = temps[1][1:]
            if temps[2][0] ==  "0"or temps[2][0] == "8":
                if temps[2][1] == "0"or temps[2][1] == "8":
                    temps[2] = temps[2][2:]
                else:
                    temps[2] = temps[2][1:]
            if temps[3][0] ==  "0"or temps[3][0] == "8":
                if temps[3][1] == "0"or temps[3][1] == "8":
                    temps[3] = temps[3][2:]
                else:
```

```
                    temps[3] = temps[3][1:]

            #reapply regular expression to string to make sure nothing slipped by.
            temps[0] = non_decimal.sub('', temps[0])
            temps[1] = non_decimal.sub('', temps[1])
            temps[2] = non_decimal.sub('', temps[2])
            temps[3] = non_decimal.sub('', temps[3])



            i = 0
            self.lastAlertOnFlag = self.alertOnFlag
            self.alertOnFlag = 0 #clear the status of the alerts
            if(self.spray_flag == 1):# continue to make sure this function doesn't
interfere with spraying by sending a spray command if spray is currently toggled on.
                #print("here")
                self.ser.write(b'a')

            for e in temps: #for each of the four temperature options
                if float(e) <= -127.0: #-127.0 is only received if there is an error
reading the temp sensor
                    self.tempSenseLabels[i].config(text = "Error reading temperature
sensor " + str(i+1)) #i+1 is the number of the temp sensor being investigated.
                    if self.lastAlertOnFlag == 1:
                        self.alertOnFlag = 1
                elif float(e) >= 35.0: #27 degrees is the level set for the warning
                    self.tempSenseLabels[i].config(text = "Temperature Sensor " +
str(i+1) + " reads " + str(e))
                    self.alertOnFlag = 1 #set the flag for warning to true
                else:
                    self.tempSenseLabels[i].config(text = "Temperature Sensor " +
str(i+1) + " reads " + str(e))

                i +=1
            if(self.spray_flag == 1):# continue to make sure this function doesn't
interfere with spraying by sending a spray command if spray is currently toggled on.
                #print("here")
                self.ser.write(b'a')
            if self.alertOnFlag==1: #if any of the sensors triggered an alert, set the
alert image to red, otherwise set it to green
                self.alertButton.config(image=self.photo2)
            else:
                self.alertButton.config(image=self.photo3)

        self.root.after(405, self.update_temp_sensors)  # run this function again in
405ms

    def start_spraying(self, event):
        if self.e_stop_flag == 0:
            self.spray_flag = 1

    def stop_spraying(self, event):
        # print("stop called")
        self.spray_flag = 0
        self.servoFlag = 0

    def move_up(self, event):
        if self.e_stop_flag == 0:
            self.servoFlag = 1

    def spray_move_up(self, event):
        if self.e_stop_flag == 0:
            self.servoFlag = 1
```

```
            self.spray_flag = 1

    def move_down(self, event):
        if self.e_stop_flag == 0:
            self.servoFlag = -1

    def spray_move_down(self, event):
        if self.e_stop_flag == 0:
            self.servoFlag = -1
            self.spray_flag = 1

    def should_send_spray(self):  # A function which is called regularly in order to
make sure that the servo receives
        # its move commands and the pump regularly receives spray commands while one
of the spray commands is held
        if self.spray_flag == 1:  # if spray held, send spray command by serial
            # print("here")
            self.ser.write(b'a')

        if self.servoFlag == 1:
            self.ser.write(b'w')
        elif self.servoFlag == -1:
            self.ser.write(b's')

        self.root.after(200, self.should_send_spray)  # this function again after 200
milliseconds

    def __init__(self):
        self.root = tk.Tk()  # Initialize the window

        self.spray_flag = 0  # make sure to not spray immediately
        self.e_stop_flag = 0  # make sure that the E-Stop doesn't start enabled.

        self.root.title("Fire Suppression Drone Spray Control Interface")  # Set the
title of the window
        self.mainframe = ttk.Frame(self.root, padding="3 3 12 12")  # set up the frame
for the left column of buttons
        self.mainframe.grid(column=0, row=0, sticky=(N, W, E, S))
        self.mainframe.columnconfigure(0, weight=1)
        self.mainframe.rowconfigure(0, weight=1)

        self.alertframe = ttk.Frame(self.root, padding="3 3 12 12")  # set up the
frame for the temperature alert image
        self.alertframe.grid(column=2, row=0, sticky=(N, W, E, S))
        self.alertframe.columnconfigure(0, weight=1)
        self.alertframe.rowconfigure(0, weight=1)

        self.alertOnFlag = 0  # set initial flags to 0
        self.lastAlertOnFlag = 0
        self.servoFlag = 0
        self.portflag = 0
        ports = list(serial.tools.list_ports.comports())
        for p in ports: #loop through all ports looking for /dev/cu.usbserial-A603UFHF
            if(p[0] == '/dev/cu.usbserial-A603UFHF'):
                self.portflag = 1
            print(p)
        if self.portflag == 0:
            self.ser = serial.Serial(str(ports[0][0]), 9600)  # initialize the serial
communication
        else:
            self.ser = serial.Serial('/dev/cu.usbserial-A603UFHF', 9600)  # initialize
the serial communication
        image = im.open("Emergency_Stop.png")  # load E-Stop button image
```

```
        photo = ImageTk.PhotoImage(image)
        label = Label(image=photo)
        label.image = photo

        image2 = im.open("redbox.png")  # load image for temp-too-high warning
        self.photo2 = ImageTk.PhotoImage(image2)
        labelalert = Label(image=self.photo2)
        labelalert.image = photo

        image3 = im.open("greenbox.png")  # load image for temp in normal range
        self.photo3 = ImageTk.PhotoImage(image3)
        labelalert2 = Label(image=self.photo3)
        labelalert2.image = photo

        self.spray_button = ttk.Button(self.mainframe, text="Spray",
command=self.on_spray_button_clicked, width=60) # create button for spraying
        self.spray_button.grid(column=2, row=0)
        self.spray_button.bind('<ButtonPress-1>',self.start_spraying)  # bind function
to start spraying
        self.spray_button.bind('<ButtonRelease-1>',self.stop_spraying)  # bind
function to stop spraying

        self.spray_up_button = ttk.Button(self.mainframe, text="Spray and rotate up",
command=self.on_spray_up_button_clicked, width=60) # create button to spray and rotate
up
        self.spray_up_button.grid(column=2, row=1)
        self.spray_up_button.bind('<ButtonPress-1>',self.spray_move_up)  # bind
function to start spraying and move servo up
        self.spray_up_button.bind('<ButtonRelease-1>',self.stop_spraying)  # bind
function to stop spraying

        self.spray_down_button = ttk.Button(self.mainframe, text="Spray and rotate
down", command=self.on_spray_down_button_clicked, width=60)  # create button to spray
and rotate down
        self.spray_down_button.grid(column=2, row=2)
        self.spray_down_button.bind('<ButtonPress-1>',self.spray_move_down)  # bind
function to start spraying and move servo down
        self.spray_down_button.bind('<ButtonRelease-1>',self.stop_spraying)  # bind
function to stop spraying

        self.down_button = ttk.Button(self.mainframe, text="rotate down",
command=self.on_down_button_clicked, width=60)  # create button to rotate down
        self.down_button.grid(column=2, row=4)
        self.down_button.bind('<ButtonPress-1>',self.move_down) #bind function to move
servo down

        self.up_button = ttk.Button(self.mainframe, text="rotate up",
command=self.on_up_button_clicked, width=60) #create button to rotate up
        self.up_button.grid(column=2, row=3)
        self.up_button.bind('<ButtonPress-1>',self.move_up)  # bind function to move
servo up

        self.stop_button = ttk.Button(self.mainframe, text="stop spraying",
command=self.on_stop_button_clicked, width=60)  # create button and bind function to
immediately stop spraying on press
        self.stop_button.grid(column=2, row=5)

        self.e_stop_button = ttk.Button(self.mainframe,
command=self.on_e_stop_button_clicked, width=60, image=photo)  # create button and
bind functionality to immediately disable full subsystem until restart of subsystem
and GUI
        self.e_stop_button.grid(column=2, row=6)
```

```
        self.alertButton = ttk.Button(self.alertframe, width=60, image=self.photo3)  #
create image for temp level warning image
        self.alertButton.grid(column=2, row=6)

        # create the labels for displaying the temperature sensor data
        self.tempSenseLabels = []
        self.tempSenseLabels.append(0)
        self.tempSenseLabels.append(0)
        self.tempSenseLabels.append(0)
        self.tempSenseLabels.append(0)
        self.tempSenseLabels[0] = ttk.Label(self.mainframe, text="temp sensor 1")
        self.tempSenseLabels[0].grid(column=2, row=7)
        self.tempSenseLabels[1] = ttk.Label(self.mainframe, text="temp sensor 2")
        self.tempSenseLabels[1].grid(column=2, row=8)
        self.tempSenseLabels[2] = ttk.Label(self.mainframe, text="temp sensor 3")
        self.tempSenseLabels[2].grid(column=2, row=9)
        self.tempSenseLabels[3] = ttk.Label(self.mainframe, text="temp sensor 4")
        self.tempSenseLabels[3].grid(column=2, row=10)

        # start polled functions
        self.root.after(100, self.should_send_spray)
        self.root.after(150, self.update_temp_sensors)

# initialize gui
gui = MQPGui()

# repeatedly loop gui to allow functions from button presses to be called.
gui.root.mainloop()
```

Appendix G

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <SoftwareSerial.h>
#include <Servo.h>



#define ONE_WIRE_BUS_PIN 12
//#define WARNING_TEMP 30.00 //at 25C for testing. change to higher temp for drone implementation

//int readTime = 250; //in ms
unsigned long lastRead = 0;

// Setup a oneWire instance to communicate with any OneWire devices
OneWire oneWire(ONE_WIRE_BUS_PIN);

// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);


// Assign the addresses of temp sensors.
DeviceAddress Probe01 = { 0x10, 0x4B, 0xEE, 0x07, 0x03, 0x08, 0x00, 0x1A };
DeviceAddress Probe02 = { 0x10, 0x14, 0xF3, 0x07, 0x03, 0x08, 0x00, 0x80 };
DeviceAddress Probe03 = { 0x10, 0x71, 0x05, 0x08, 0x03, 0x08, 0x00, 0x08 };
DeviceAddress Probe04 = { 0x10, 0xDA, 0xE1, 0x07, 0x03, 0x08, 0x00, 0xF3 };
/* new addresses, to test which to replace
 *  0x10, 0xC3, 0xEA, 0x07, 0x03, 0x08, 0x00, 0x4E
 *  0x10, 0xB6, 0xD5, 0x07, 0x03, 0x08, 0x00, 0xD6

 */


int sprayPin = 13;      // LED connected to digital pin 13
Servo rotServo;
//set up variables for state tracing
int flag = 0;
int servoFlag = 0;
int lastServoFlag = 0;
unsigned long timeOfLastA = 0;
unsigned long timeOfLastServoSignal = 0;
int eStopFlag = 0;
//set up software serial connection
SoftwareSerial mySerial(10, 11); // RX, TX
char readChar = 0;
void setup() {
  // Initialize the Temperature measurement library
  sensors.begin();

  // set the resolution to 10 bit (Can be 9 to 12 bits .. lower is faster)
  sensors.setResolution(Probe01, 9);
```

```
  sensors.setResolution(Probe02, 9);
  sensors.setResolution(Probe03, 9);
  sensors.setResolution(Probe04, 9);

  //Attach the servo for spray direction and set it to starting position.
  rotServo.attach(8);
  rotServo.write(0);
  pinMode(sprayPin, OUTPUT);   // sets the pin as output
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {

  }


  // set the data rate for the SoftwareSerial port
  mySerial.begin(9600);
}

void loop() { // run over and over
  delay(10);

  if (eStopFlag == 0) {
    //Serial.println("here");
    if (mySerial.available()) {
      readChar = mySerial.read();
      //Serial.println(readChar);
      if (readChar == 97) {
        flag =  1;
        timeOfLastA = millis();
        //Serial.println(readChar);
      }
      else if (readChar == 98) {
        flag = 0;
        timeOfLastA = 0;
        analogWrite(sprayPin, 0);
        //Serial.println(readChar);
      }
      else if (readChar == 104) {
        eStopFlag = 1;
        flag = 0;
        //Serial.println("ESTOP");
        analogWrite(sprayPin, 0);
      }
      else if (readChar == 119) {
        servoFlag = 1;

        //Serial.println(readChar);
      }
      else if (readChar == 115) {
```

```
    servoFlag = -1;
    //Serial.println(readChar);
   }
  }
  if (flag == 1 && eStopFlag == 0) {
   if ((millis() - timeOfLastA) < 3000) {
    analogWrite(sprayPin, 225);  // analogRead values go from 0 to 1023, analogWrite values from 0 to 255
   }
   else {
    flag = 0;
    analogWrite(sprayPin, 0);
   }


   //else Serial.println(readChar);
   //Serial.println(mySerial.read());
  }
  if (servoFlag == 1) {
   rotServo.write(10);
   servoFlag = 0;
   lastServoFlag = 1;
  }
  else if (servoFlag == -1) {
   rotServo.write(80);
   servoFlag = 0;
  }
  else if (servoFlag != 0) {
   //Serial.println(servoFlag);
  }
  if (Serial.available()) {
   mySerial.println(Serial.read());
  }
}
/*Serial.print("LastRead = ");
Serial.print(lastRead);
Serial.print(" millis = ");
Serial.print(millis());
Serial.println();*/
if ( (millis() - 1000 >= lastRead) && millis() > 1000) {

 //Serial.println("here");
 // Command all devices on bus to read temperature
 lastRead = millis();
 sensors.requestTemperatures(); //bring this back

 //mySerial.print("Probe 01 temperature is:   ");
 //Serial.println("temp sensors");
 printTemperature(Probe01); //bring this back
```

```
   //mySerial.print("Probe 02 temperature is:   ");
   printTemperature(Probe02);// bring this back


   //mySerial.print("Probe 03 temperature is:   ");
   printTemperature(Probe03); //bring this back


   //mySerial.print("Probe 04 temperature is:   ");
   printTemperature(Probe04); //bring this back
   mySerial.println();
   //mySerial.println();
  }
  delay(10);
}
void printTemperature(DeviceAddress deviceAddress)
{
  float tempC = sensors.getTempC(deviceAddress);

  //if (tempC == -127.00)
  //{
  //mySerial.print("Error getting temperature  ");
  //}
  //else if (tempC >=  WARNING_TEMP)
  //{
  //mySerial.print("WARNING");
  //mySerial.println();

  //mySerial.print("C: %f F: %F", tempC, DallasTemperature::toFahrenheit(tempC));
  //mySerial.print("C: ");
  //mySerial.print(tempC);
  //mySerial.print(" F: ");
  //mySerial.print(DallasTemperature::toFahrenheit(tempC));
  //mySerial.print(printString);
  //}
  //else {
  //String printString =
  //mySerial.print("C: %f F: %F", tempC, DallasTemperature::toFahrenheit(tempC));
  //mySerial.print("C: ");
  if(tempC>=0){
    mySerial.print("0");
  }
  if(tempC>100){
    mySerial.print("0");

  }
  if(tempC>10){
    mySerial.print("0");
  }
  mySerial.print(tempC); //bring this back
  //Serial.println(tempC);
```

```
  //mySerial.print(" F: ");
  //mySerial.print(DallasTemperature::toFahrenheit(tempC));
  //mySerial.print(printString);
  //}
}
```

Bibliography

"Fire Safety on Boats." Communities and Local Government, 2012. Print.

World Fire Statistics: The Geneva Association, 2010. Print.

Beard, Randal, and Timothy McLain. *Small Unmanned Aircraft: Theory and Practice*. Princeton, NJ, USA: Princeton University Press, 2012. Print.

Biesner, Eryn et al. "Acoustic Flame Suppression Mechanics in a Microgravity Environment." Springer Science and Business Media, 2015. Print.

Brauer, Molly. "Pump up the Bass to Douse a Blaze: Mason Students' Invention Fights Fires." *George Mason University News* 2015. Print.

Chu, Jennifer. "Projecting a Robot's Intentions." 2014. Web.

Connection, Fire Department. "Class K Fire Extinguishers " 2012. Print.

Conroy, Mark. "Nfpa Fire Protection Handbook: Fire Extinguisher Use and Maintenance." Quincy, MA: NFPA, 2008. Print.

DARPA. *Darpa Instant Flame Suppression Phase Ii Final Report: Acoustic Waves*: Harvard University, 2008. Print.

---. *Instant Flame Suppression Phase Ii Final Report: Electrostatics*: Harvard University, 2008. Print.

Department, Rocky Mountain Fire. "Kitchen Grease Fire." Web.

Dinenno, Philip, and Gary Taylor. "Nfpa Fire Protection Handbook: Halon and Halon Replacement Agents and Systems." Quincy, MA: NFPA, 2008. Print.

Drysdale, Dougal. "An Introduction to Fire Dynamics." 2nd ed. West Sussex, England: John Wiley & Sons Ltd, 1999. Print.

Englert, Ken. "Is Your Boat an Electrical Fire Hazard?" Boating *2013*. Web. October 3 2015.

EPRI. "Fire Probablistic Risk Assessment Methods Enhancements: Supplements 1 to Nureg/Cr-6850 and Epri 1011989." Palo Alto, CA and Washington, DC: EPRI and NRC, 2009. Print.

FEMA. "Fire Death Rate Trends: An International Perspective." FEMA2011. Print.

---. "U.S. Fire Statistics." 2015. Web. 9/27/2015 2015.

Floreano, Dario, and Robert J. Wood. "Science, Technology and the Future of Small Autonomous Drones." *Nature* 521.7553 (2015): 460-66. Print.

GelTech Solutions, Inc. "Geltech Solutions 2015 Wildland Fire Season Review, 2016 Growth Projections." 2016. Print.

---. "Geltech's Fireice Gaining Acceptance by Fire Departments, Police Departments, Hazmat Teams and Colleges across the Northeast." 2016. Print.

Hamins, A., and Nist Engineering Laboratory Building and Fire Publications. *Vehicle Fire Suppression Research Needs*. 2007. Print.

Hamins, Anthony. *Evaluation of Active Suppression on Simulated Post-Collision Vehicle Fire*: U.S. Department of Commerce, 2001. Print.

Haynes, Hylton. "Fire Loss in the United States." National Fire Protection Agency, 2015. Print.

Inc, GelTech Solutions. "Eco-Friendly Fireice Used to Combat 100+ Wildfires (and Counting) in the U.S. And Canada This Season." PRNewswire.com: PRNewswire, 2015. Print.

---. "Geltech's Fireice Continues to Impress This Wildland Fire Season." PRNewswire.com: PRNewswire, 2015. Print.

Kennedy, Gerry. "Interview with Gerry Kennedy." Ed. Team, Fire Containment Drone2015. Print.

Laboratories, Underwriters. "Ul Company Fact Sheet." Print.

Lake, James. "Nfpa Fire Protection Handbook: Chemical Extinguishing Agents and Application Systems." Quincy, MA: NFPA, 2008. Print.

Leonard, Beth. "Why Boats Catch Fire." Boat Owners Association of The United States 2015. Web.

Madryzkowski, Dan. "Fire Dynamics." NIST 2013. Web. 3/26/2016 2016.

Management, US Forest Service Fire and Aviation. "Water Enhancers (Gels) for Fire Management." *Qualified By US Forest Service in Accordance with Forest Service Specification 5100-306A as Amended*: US Forest Service Fire and Aviation Management, 2015. Print.

Matros, Mike. "Interview with Mike Matros." Ed. Team, Fire Containment Drone2015. Print.

NASA. "Aircraft Rotations Body Axes." 2015. Print.

---. "What Is a Helicopter." 2014. Web. 10/4/2015 2015.

National Fire Protection Association., Society of Fire Protection Engineers., and Books24x7 EngineeringPro Collection. *Sfpe Handbook of Fire Protection Engineering*. 2002.Web

National Research Council, Staff, Complete Ebrary Academic, and Halon National Research Council . Naval Studies Board.Committee on Assessment of Fire Suppression Substitutes and Alternatives to. *Fire Suppression Substitutes and Alternatives to Halon for U.S. Navy Applications*. Washington, D.C: National Academy Press, 1997. Print.

NFPA. "All About Fire." National Fire Protection Association 2015. Web. 9/24/2015 2015.

Nir, Sarah Maslin. "*California Fire, Aided by Drought, Defies Tactics to Defeat It*."*New York Times* 2015. Print.

NIST. "Fire Dynamics." 2013. Web.

Norman, John. *Fire Officer's Handbook of Tactics*. 4th ed. Tulsa, Oklahoma: PennWell Corporation, 2012. Print.

Roberts, Mary Rose. "5 Drone Technologies for Firefighting." FireRescue 2014. Web2015.

Rosenblum, Yael Student author R. B. E., Brian William Student author R. B. E. Gallenstein, and Yiming Faculty advisor M. E. Rong. *Autonomous Fixed-Point Landing for Quadrotor Aerial Vehicles*. Worcester, MA U6 eBook: Worcester Polytechnic Institute, 2015. Print.

Sargent, Jamie. "The Scoop on Water Bombers." Fire Boss LLC, 2011. Print.

Scheffey, Joseph. "Nfpa Fire Protection Handbook: Foam Extinguishing Agents and Systems." Quincy, MA: NFPA, 2008. Print.

Sivak, John Joseph Student author R. B. E., et al. *Quadrotor Uav*. Worcester, MA U6 eBook: Worcester Polytechnic Institute, 2011. Print.

Telleria, Mike. "Boat Fires – Prevention, Suppression, Detection & Preparedness." Print.

WHO. *Global Health Observatory Data Repository: Mortality and Global Health Estimates*2012. Print.

Wikimedia. "Fire Triangle." Print.

Wysocki, Thomas. "Nfpa Fire Protection Handbook: Carbon Dioxide and Application Systems." Quincy, MA: NFPA, 2008. Print.

Qgroundcontrol, “Mavlink Packet” http://qgroundcontrol.org/_media/mavlink/mavlink-packet.png?cache=